

Symt

10C9
10-
10-
10-
10-
10-
10-
KICL

KILL
KILL
LB_E
LB_C
LB_F
LB_H
LB_L
LOCA
LOCA
LOCK

LOCK
LOCK
LOCK
LOC-
LOC-
L-CC
L-CC
L-DA
L-DA
MAIN
MAKE
MAKE
MAKE
MAKE
MAKE
MAKE

MAKE
MAKE
MAP
MAP

MAP
MAR
MAR
MAR
MAR
MAR

FFFFFFFFFFFFFFFF	111	111	XXX	XXX
FFFFFFFFFFFFFFFF	111	111	XXX	XXX
FFFFFFFFFFFFFFFF	111	111	XXX	XXX
FFF	111111	111111	XXX	XXX
FFF	111111	111111	XXX	XXX
FFF	111111	111111	XXX	XXX
FFF	111	111		
FFF	111	111	XXX	XXX
FFF	111	111	XXX	XXX
FFFFFFFFF.FFF	111	111		
FFFFFFFFFFFFFFFF	111	111	XXX	
FFFFFFFFFFFFFFFF	111	111	XXX	
FFF	111	111		
FFF	111	111	XXX	XXX
FFF	111	111	XXX	XXX
FFF	111	111	XXX	XXX
FFF	111	111		
FFF	111	111	XXX	XXX
FFF	111	111	XXX	XXX
FFF	111	111	XXX	XXX
FFF	1111111111	1111111111	XXX	XXX
FFF	1111111111	1111111111	XXX	XXX
FFF	1111111111	1111111111	XXX	XXX

[illegible]

```

LL          IIIIII          SSSSSSSS
LL          IIIIII          SSSSSSSS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SSSSSS
LL          II             SSSSSS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SS
LLLLLLLLLLLL IIIIII          SSSSSSSS
LLLLLLLLLLLL IIIIII          SSSSSSSS

```



```
1 0001 0 MODULE CLENUP (
2 0002 0 LANGUAGE (BLISS32),
3 0003 0 IDENT = 'V04-000'
4 0004 0 ) =
5 0005 1 BEGIN
6 0006 1
7 0007 1
8 0008 1 *****
9 0009 1 *
10 0010 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
11 0011 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
12 0012 1 * ALL RIGHTS RESERVED.
13 0013 1 *
14 0014 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
15 0015 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
16 0016 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
17 0017 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
18 0018 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
19 0019 1 * TRANSFERRED.
20 0020 1 *
21 0021 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
22 0022 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
23 0023 1 * CORPORATION.
24 0024 1 *
25 0025 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
26 0026 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
27 0027 1 *
28 0028 1 *
29 0029 1 *****
30 0030 1
31 0031 1 ++
32 0032 1
33 0033 1 FACILITY: F11ACP Structure Level 2
34 0034 1
35 0035 1 ABSTRACT:
36 0036 1
37 0037 1 This module performs the necessary cleanup after an operation.
38 0038 1
39 0039 1 ENVIRONMENT:
40 0040 1
41 0041 1 STARLET operating system, including privileged system services
42 0042 1 and internal exec routines.
43 0043 1
44 0044 1 --
45 0045 1
46 0046 1
47 0047 1 AUTHOR: Andrew C. Goldstein, CREATION DATE: 6-Jan-1977 23:53
48 0048 1
49 0049 1 MODIFIED BY:
50 0050 1
51 0051 1 V03-034 CDS0022 Christian D. Saether 30-Aug-1984
52 0052 1 Allow for multi-header directory files.
53 0053 1 Have error cleanup remove possible bias on primary_fcb
54 0054 1 refcnt.
55 0055 1
56 0056 1 V03-033 CDS0021 Christian D. Saether 23-Aug-1984
57 0057 1 Move code that marks FCB stale to a routine in LOCKERS.
```

58	0058	1	V03-032	CDS0020	Christian D. Saether	13-Aug-1984	
59	0059	1		Add code to mark primary fcb stale clusterwide.			
60	0060	1					
61	0061	1					
62	0062	1	V03-031	CDS0019	Christian D. Saether	7-Aug-1984	
63	0063	1		Cleanup potential directory index cache block			
64	0064	1		when deleting a file.			
65	0065	1					
66	0066	1	V03-030	CDS0018	Christian D. Saether	1-Aug-1984	
67	0067	1		Modify test for directory fcb.			
68	0068	1		Add SET DIRINDX routine.			
69	0069	1		Add NUKE PRIM FCB routine.			
70	0070	1		Modify ZERO_IDX routine.			
71	0071	1					
72	0072	1	V03-029	ACG0438	Andrew C. Goldstein,	19-Jul-1984	17:55
73	0073	1		Add cluster-wide special cache interlock logic.			
74	0074	1		Condition DELETEACL calls on non-empty ACL.			
75	0075	1		Use central dequeue routine.			
76	0076	1					
77	0077	1	V03-028	CDS0017	Christian D. Saether	25-May-1984	
78	0078	1		Call KILL_BUFFERS routine to flush cache in			
79	0079	1		certain situations when not in a cluster.			
80	0080	1					
81	0081	1	V03-027	CDS0016	Christian D. Saether	9-May-1984	
82	0082	1		Release allocation lock prior to calling send_symbiont.			
83	0083	1					
84	0084	1	V03-026	CDS0015	Christian D. Saether	4-May-1984	
85	0085	1		No not map notrunc into nowrite.			
86	0086	1		Add bugcheck if access lock conversion fails in make_deaccess.			
87	0087	1					
88	0088	1	V03-025	CDS0014	Christian D. Saether	3-May-1984	
89	0089	1		Call CONV_ACCLOCK to remove possible access lock			
90	0090	1		when deallocating fcb's.			
91	0091	1					
92	0092	1	V03-024	CDS0013	Christian D. Saether	19-Apr-1984	
93	0093	1		Changes to FCB\$W_ACNT handling.			
94	0094	1					
95	0095	1	V03-023	ACG0415	Andrew C. Goldstein,	5-Apr-1984	21:27
96	0096	1		Interface change to ACL_DELETEACL			
97	0097	1					
98	0098	1	V03-022	ACG0408	Andrew C. Goldstein,	23-Mar-1984	11:20
99	0099	1		Make rest of global storage based			
100	0100	1					
101	0101	1	V03-021	CDS0012	Christian D. Saether	9-Mar-1984	
102	0102	1		Put in bug trap to catch possible double remque of			
103	0103	1		FCB.			
104	0104	1					
105	0105	1	V03-020	CDS0011	Christian D. Saether	23-Feb-1984	
106	0106	1		Use new WRITE_DIRTY routine to replace FLUSH_BUFFERS.			
107	0107	1		Remove references to FLUSH_FID.			
108	0108	1		Replace FLUSH_FID (0) with KILL_CACHE calls.			
109	0109	1					
110	0110	1	V03-019	CDS0010	Christian D. Saether	27-Dec-1983	
111	0111	1		Use L_NORM linkage.			
112	0112	1		Use BIND_COMMON macro to reduce external declarations.			
113	0113	1					
114	0114	1	V03-018	CDS0009	Christian D. Saether	23-Nov-1983	

115	0115	1	If DIR_FCB is the same as PRIMARY_FCB, do not return the FCB until the end of cleanup (as PRIMARY_FCB, not DIR_FCB). Move cleanup of DIR_FCB until after all i/o is done. Remove REMOVE_FCB routine (kernel call not necessary).
116	0116	1	
117	0117	1	
118	0118	1	
119	0119	1	
120	0120	1	
121	0121	1	V03-017 LMP0164 L. Mark Pilant, 10-Oct-1983 15:22
122	0122	1	Delete the in-core ACL if doing an FCB fixup.
123	0123	1	
124	0124	1	V03-016 CDS0008 Christian D. Saether 3-Oct-1983
125	0125	1	Handle CURR_LCKINDX in err_cleanup. Don't read
126	0126	1	headers without appropriate serial locks.
127	0127	1	
128	0128	1	V03-015 CDS0007 Christian D. Saether 14-Sep-1983
129	0129	1	Take out deqall hack now that RMS does it's own
130	0130	1	root locks again.
131	0131	1	
132	0132	1	V03-014 CDS0006 Christian D. Saether 27-Jul-1983
133	0133	1	Change interface to SEND_SYMBIONT.
134	0134	1	
135	0135	1	V03-013 LJK0199 Lawrence J. Kenah 27-Apr-1983
136	0136	1	Do not credit FILCNT when giving back shared window
137	0137	1	
138	0138	1	V03-012 CDS0006 Christian D. Saether 28-Apr-1983
139	0139	1	Clear DIR_ENTRY when DIR_FCB is cleared.
140	0140	1	
141	0141	1	V03-011 CDS0005 Christian D. Saether 21-Apr-1983
142	0142	1	Change interface to TRUNCATE routine.
143	0143	1	
144	0144	1	V03-010 CDS0004 Christian D. Saether 19-Apr-1983
145	0145	1	Bug check on unexpected lock manager errors.
146	0146	1	Clear ACCLKID field in window.
147	0147	1	
148	0148	1	V03-009 ACG0323 Andrew C. Goldstein, 12-Apr-1983 14:09
149	0149	1	Add extended file name to back link fixup
150	0150	1	
151	0151	1	V03-008 STJ3069 Steven T. Jeffreys, 23-Mar-1983
152	0152	1	Use the ERASE_REQUESTED parameter of RETURN_BLOCKS.
153	0153	1	
154	0154	1	V03-007 CDS0003 Christian D. Saether 7-Mar-1983
155	0155	1	Perform a DEQALL if file access lock dequeue fails
156	0156	1	due to sublocks, then redo the file access dequeue.
157	0157	1	
158	0158	1	V03-006 LMP0071 L. Mark Pilant, 19-Jan-1983 20:49
159	0159	1	Correct a problem that caused ACL segments to be left laying
160	0160	1	around when a directory FCB was flushed.
161	0161	1	
162	0162	1	V03-005 ACG0308 Andrew C. Goldstein, 14-Jan-1983 15:02
163	0163	1	Fix FCB linkage consistency problems
164	0164	1	
165	0165	1	V03-004 CDS0002 Christian D. Saether 3-Jan-1983
166	0166	1	Always flush header cache until it is restored for xqp.
167	0167	1	
168	0168	1	V03-003 LMP0059 L. Mark Pilant, 21-Dec-1982 12:23
169	0169	1	Always create an FCB when accessing a file header. This
170	0170	1	eliminates a lot of special case FCB handling.
171	0171	1	

```

: 172 0172 1 : V03-002 CDS0001 Christian D. Saether 10-Dec-1982
: 173 0173 1 : MAKE_DEACCESS dequeues access lock.
: 174 0174 1 :
: 175 0175 1 : V03-001 LMP0036 L. Mark Pilant, 17-Aug-1982 10:45
: 176 0176 1 : If the ACL was built using a dummy FCB, dismantle and
: 177 0177 1 : deallocate the ACL.
: 178 0178 1 :
: 179 0179 1 : V02-024 ACG0259 Andrew C. Goldstein, 26-Jan-1982 19:12
: 180 0180 1 : Add mode arg to REMOVE
: 181 0181 1 :
: 182 0182 1 : V02-023 ACG0247 Andrew C. Goldstein, 23-Dec-1981 20:26
: 183 0183 1 : Make /NOCACHE flush all caches
: 184 0184 1 :
: 185 0185 1 : V02-022 ACG0245 Andrew C. Goldstein, 23-Dec-1981 20:26
: 186 0186 1 : Send spool file to print during cleanup
: 187 0187 1 :
: 188 0188 1 : V02-021 ACG0244 Andrew C. Goldstein, 23-Dec-1981 20:14
: 189 0189 1 : Do buffer flush before deallocating control blocks
: 190 0190 1 :
: 191 0191 1 : V02-020 LMP0003 L. Mark Pilant, 30-Nov-1981 16:40
: 192 0192 1 : Properly cleanup any cathedral windows.
: 193 0193 1 :
: 194 0194 1 : V02-019 ACG0208 Andrew C. Goldstein, 11-Nov-1981 17:51
: 195 0195 1 : Add segmented directory record support
: 196 0196 1 :
: 197 0197 1 : V02-018 ACG0168 Andrew C. Goldstein, 7-May-1980 18:22
: 198 0198 1 : Fix last block directory cleanup on delete failure
: 199 0199 1 :
: 200 0200 1 : V02-017 ACG0167 Andrew C. Goldstein, 16-Apr-1980 19:25
: 201 0201 1 : Previous revision history moved to F11B.REV
: 202 0202 1 : **
: 203 0203 1 :
: 204 0204 1 :
: 205 0205 1 : LIBRARY 'SYSS$LIBRARY:LIB.L32';
: 206 0206 1 : REQUIRE 'SRC$FCPDEF.B32';
: 207 1197 1 :
: 208 1198 1 :
: 209 1199 1 : FORWARD ROUTINE
: 210 1200 1 : CLEANUP : L_NORM, : normal cleanup
: 211 1201 1 : ZERO_WINDOWS : L_NORM, : invalidate all windows of file
: 212 1202 1 : ZERO_IDX : L_NORM NOVALUE, : initialize directory index
: 213 1203 1 : ERR_CLEANUP : L_NORM, : cleanup after error
: 214 1204 1 : FLUSH_FIDCACHE : L_NORM, : clean out the file ID cache
: 215 1205 1 : MAKE_DEACCESS : L_NORM, : deaccess the file
: 216 1206 1 : DEL_EXTFCB : L_NORM, : deallocate extension FCB's
: 217 1207 1 : ZERO_CHANNEL : L_NORM, : zero user channel pointer
: 218 1208 1 : SET_DIRINDX : L_JSB 1ARG, : test for directory index
: 219 1209 1 : NUKE_HEAD_FCB : L_NORM NOVALUE, : deallocate primary fcb
```



```
221 1210 1 GLOBAL ROUTINE CLEANUP : L_NORM =
222 1211 1
223 1212 1 ++
224 1213 1
225 1214 1 FUNCTIONAL DESCRIPTION:
226 1215 1
227 1216 1 This routine performs the cleanup needed after a successfully
228 1217 1 completed file operation.
229 1218 1
230 1219 1 CALLING SEQUENCE:
231 1220 1 CLEANUP ()
232 1221 1
233 1222 1 INPUT PARAMETERS:
234 1223 1 NONE
235 1224 1
236 1225 1 IMPLICIT INPUTS:
237 1226 1 CLEANUP_FLAGS: indicate specific actions to do
238 1227 1 PRIMARY_FCB: FCB of file
239 1228 1 CURRENT_WINDOW: window of file
240 1229 1 DIR_FCB: FCB of directory
241 1230 1 CURRENT_VCB: VCB of volume in process
242 1231 1 IO_PACKET: I/O packet of request
243 1232 1
244 1233 1 OUTPUT PARAMETERS:
245 1234 1 NONE
246 1235 1
247 1236 1 IMPLICIT OUTPUTS:
248 1237 1 NONE
249 1238 1
250 1239 1 ROUTINE VALUE:
251 1240 1 NONE
252 1241 1
253 1242 1 SIDE EFFECTS:
254 1243 1 FCB's and windows deleted when appropriate
255 1244 1 header written
256 1245 1 FCB updated
257 1246 1
258 1247 1 --
259 1248 1
260 1249 2 BEGIN
261 1250 2
262 1251 2 LOCAL
263 1252 2 CLUSTER,          ! are we a cluster
264 1253 2 QUOTA_CACHE    : REF BBLOCK, ! address of quota cache
265 1254 2 FCB           : REF BBLOCK, ! local FCB pointer
266 1255 2 VCB           : REF BBLOCK, ! local VCB pointer
267 1256 2 RVT          : REF BBLOCK, ! local RVT pointer
268 1257 2 UCB          : REF BBLOCK, ! local UCB pointer
269 1258 2 HEADER        : REF BBLOCK; ! file header
270 1259 2
271 1260 2 BIND_COMMON;
272 1261 2
273 1262 2 DIR_CONTEXT_DEF;
274 1263 2
275 1264 2 EXTERNAL
276 1265 2 CLUSGL_CLUB      : ADDRESSING_MODE (ABSOLUTE);
277 1266 2
```

```
278 1267 2 EXTERNAL ROUTINE
279 1268 2 MAKE_FCB_STALE : L_NORM NOVALUE, ! mark fcb as stale clusterwide
280 1269 2 KILL_BUFFERS : L_NORM NOVALUE, ! invalidate specified buffers
281 1270 2 KILL_CACHE : L_NORM NOVALUE, ! invalidate all buffers for ucb
282 1271 2 WRITE_DIRTY : L_NORM, ! write all dirty buffers
283 1272 2 SWITCH_VOLUME : L_NORM, ! switch to desired volume
284 1273 2 FLUSH_QUO_CACHE : L_NORM; ! flush the quota cache
285 1274 2
286 1275 2
287 1276 2 ! ***** Note: The primary header of the current file is not necessarily
288 1277 2 ! resident at this point.
289 1278 2
290 1279 2 ! Switch back to the primary context area if necessary (no normal cleanup
291 1280 2 ! is ever necessary on secondary context).
292 1281 2 !
293 1282 2
294 1283 2 IF .CONTEXT_SAVE NEQ 0
295 1284 2 THEN
296 1285 2 BEGIN
297 1286 2 CH$MOVE (CONTEXT_SIZE, CONTEXT_SAVE, CONTEXT_START);
298 1287 2 CONTEXT_SAVE = 0;
299 1288 2 END;
300 1289 2
301 1290 2 CLUSTER = 0;
302 1291 2 IF .BBLOCK [CURRENT_UCB [UCB$L_DEVCHAR2], DEV$V_CLU]
303 1292 2 AND .CLUSGL_CLUB NEQ 0
304 1293 2 THEN
305 1294 2 CLUSTER = 1;
306 1295 2
307 1296 2 ! Check the entire volume set to see if the index file or storage map
308 1297 2 ! on any volume is write accessed. If so, flush the buffer pool of any
309 1298 2 ! of their blocks, and flush the file ID and extent caches as appropriate.
310 1299 2 ! Also, if the volume is mounted /NOCACHE, flush the entire buffer cache.
311 1300 2 !
312 1301 2
313 1302 2 RVT = .CURRENT_VCB[VCB$L_RVT];
314 1303 2 INCR J FROM 1 TO
315 1304 2 BEGIN
316 1305 2 IF .RVT EQL .CURRENT_UCB
317 1306 2 THEN (UCB = .RVT; 1)
318 1307 2 ELSE .RVT[RVT$B_NVOLS]
319 1308 2 END
320 1309 2 DO
321 1310 2 BEGIN
322 1311 2 IF .RVT NEQ .CURRENT_UCB
323 1312 2 THEN UCB = .VECTOR [RVT[RVT$L_UCBLST], .J-1];
324 1313 2 IF .UCB NEQ 0
325 1314 2
326 1315 2 THEN
327 1316 2 BEGIN
328 1317 2 VCB = .UCB[UCB$L_VCB];
329 1318 2
330 1319 2 IF .J EQL 1
331 1320 2 THEN
332 1321 2 BEGIN
333 1322 2
334 1323 2 ! If someone has the quota file write accessed (and it is active), flush it
```



```

335 1324 5 ! from the buffer pool. (Note that the quota file is located on RVN 1.)
336 1325 5 !
337 1326 5
338 1327 5 QUOTA_CACHE = .VCB[VCB$L_QUOCACHE];
339 1328 5 IF .QUOTA_CACHE NEQ 0
340 1329 5 THEN
341 1330 5 IF TESTBITSC (QUOTA_CACHE[VCA$V_CACHEFLUSH])
342 1331 5 THEN
343 1332 6 BEGIN
344 1333 6 SWITCH_VOLUME (1);
345 1334 6 FLUSH_QUO_CACHE (); ! may create modified buffers
346 1335 5 END;
347 1336 4 END; ! of this is RVN 1 (or single volume)
348 1337 4
349 1338 4 ! If the volume is marked for dismount or nocache, flush out all the
350 1339 4 caches.
351 1340 4 !
352 1341 4
353 1342 4 IF .BBLOCK [UCB [UCB$L_DEVCHAR], DEV$V_DMT]
354 1343 4 OR .VCB[VCB$V_NOCACHE]
355 1344 4 THEN
356 1345 5 BEGIN
357 1346 5 SWITCH_VOLUME (.J);
358 1347 5 WRITE_DIRTY (0);
359 1348 5 KILL_CACHE (.UCB); ! we cannot use the block cache after this
360 1349 4 END;
361 1350 3 END;
362 1351 2 END;
363 1352 2
364 1353 2 ! Write modified buffers. The various cache purges above may have
365 1354 2 created more dirty buffers than we had at the start of this routine.
366 1355 2 ! No more dirty buffers can be created for the remainder of this request.
367 1356 2 !
368 1357 2
369 1358 2 WRITE_DIRTY (0);
370 1359 2
371 1360 2 ! Invalidate any windows on the file, if requested.
372 1361 2 !
373 1362 2
374 1363 2 IF TESTBITSC (CLEANUP_FLAGS[CLF_INVWINDOW])
375 1364 2 THEN KERNEL_CALL (ZERO_WINDOWS, ".PRIMARY_FCB");
376 1365 2
377 1366 2 ! If a directory fcb is left lying about with no use, dispose of it.
378 1367 2 ! If the directory file is write accessed, flush the buffer pool of any
379 1368 2 blocks that might be resident. Also flush the directory index.
380 1369 2 ! Cleanup of these fcb's is deferred until all possible errors in the
381 1370 2 cleanup procedure (i/o errors) have already had an opportunity to happen.
382 1371 2 !
383 1372 2
384 1373 2 IF (FCB = .DIR_FCB) NEQ 0
385 1374 2 THEN
386 1375 3 BEGIN
387 1376 3 IF .FCB [FCB$W_REFCNT] EQL 0
388 1377 3 THEN
389 1378 4 BEGIN
390 1379 4 IF .FCB NEQ .PRIMARY_FCB
391 1380 4 THEN
```



```

392      1381      4      IF NOT SET_DIRINDX (.FCB)
393      1382      4      THEN
394      1383      5      BEGIN
395      1384      5      DEL_EXTFCB (.FCB);
396      1385      5      NUKE_HEAD_FCB (.FCB);
397      1386      4      END;
398      1387      4
399      1388      4      END
400      1389      4
401      1390      3      ELSE
402      1391      4      BEGIN
403      1392      4      IF .FCB [FCB$W_WCNT] NEQ 0
404      1393      4      THEN
405      1394      5      BEGIN
406      1395      5      SWITCH_VOLUME (.FCB [FCB$W_FID_RVN]);
407      1396      5      IF NOT .CLUSTER
408      1397      5      THEN
409      1398      5      KILL_BUFFERS (1, .FCB [FCB$L_LOCKBASIS]);
410      1399      5      ZERO_IDX ();
411      1400      4      END;
412      1401      3      END;
413      1402      3
414      1403      3      ! Guarantee that no further attempts will be made to do any directory
415      1404      3      ! related cleanup. This cleanup code was moved beyond the buffer
416      1405      3      ! cleanup to avoid the same situation, but clearing the cleanup flags
417      1406      3      ! makes sure.
418      1407      3
419      1408      3
420      1409      3      CLEANUP_FLAGS [CLF_SUPERSEDE] = 0;
421      1410      3      CLEANUP_FLAGS [CLF_REENTER] = 0;
422      1411      3      CLEANUP_FLAGS [CLF_REMOVE] = 0;
423      1412      3      DIR_FCB = 0;
424      1413      3      DIR_ENTRY = 0;
425      1414      3
426      1415      3      END;
427      1416      3
428      1417      3      IF (FCB = .PRIMARY_FCB) NEQ 0
429      1418      3      THEN
430      1419      3      BEGIN
431      1420      3
432      1421      3      ! Check if the fcb has been modified and if so, and this is a cluster,
433      1422      3      ! cause potential fcb's on other nodes to be marked as stale so they
434      1423      3      ! will know to rebuild their fcb chains from the file header(s).
435      1424      3
436      1425      3
437      1426      3      IF .CLEANUP_FLAGS [CLF_MARKFCBSTALE]
438      1427      3      AND .CLUSTER
439      1428      3      THEN
440      1429      3      MAKE_FCB_STALE (.FCB);
441      1430      3
442      1431      3      ! If an FCB is left about with no use, dispose of it.
443      1432      3      ! Check whether it is a directory fcb first.
444      1433      3
445      1434      3
446      1435      3      IF .FCB[FCB$W_REFCNT] EQL 0
447      1436      3      THEN
448      1437      3      IF NOT SET_DIRINDX (.FCB)
```



```

449      1438 3      THEN
450      1439 4      BEGIN
451      1440 4
452      1441 4      DEL_EXTFCB (.FCB);
453      1442 4
454      1443 4      NUKE_HEAD_FCB (.FCB);
455      1444 4
456      1445 4      PRIMARY_FCB = 0;
457      1446 3      END;
458      1447 2      END;
459      1448 2
460      1449 2      RETURN 1;
461      1450 2
462      1451 1      END;

```

```
! end of routine CLEANUP
```

.TITLE	CLENUMP	
.IDENT	\V04-000\	
.EXTRN	CLUS\$GL CLUB, MAKE_FCB_STALE	
.EXTRN	KILL_BUFFERS, KILL_CACHE	
.EXTRN	WRITE_DIRTY, SWITCH_VOLUME	
.EXTRN	FLUSH_QUO_CACHE	
.PSECT	\$CODE\$,NOWRT,2	
.ENTRY	CLEANUP, Save R2,R3,R4,R5,R6,R7,R8,R9,R11	: 1210
MOVAB	SWITCH_VOLUME, R11	: :
MOVAB	220(BASE), R8	: 1258
TSTL	54(BASE)	: 1283
BEQL	1\$: :
MOV3	#54, 54(BASE), (BASE)	: 1286
CLRL	54(BASE)	: 1287
CLRL	CLUSTER	: 1290
MOVL	-108(BASE), R0	: 1291
BLBC	60(R0), 2\$: :
TSTL	a#CLUS\$GL CLUB	: 1292
BEQL	2\$: :
MOVL	#1, CLUSTER	: 1294
MOVL	-104(BASE), R0	: 1302
MOVL	32(R0), RVT	: :
CMPL	RVT, -108(BASE)	: 1305
BNEQ	3\$: :
MOVL	RVT, UCB	: 1306
MOVL	#1, R7	: :
BRB	4\$: :
MOVZBL	11(RVT), R7	: 1307
CLRL	J	: 1303
BRB	9\$: :
CMPL	RVT, -108(BASE)	: 1311
BEQL	6\$: :
MOVL	64(RVT)[J], UCB	: 1312
TSTL	UCB	: 1313
BEQL	9\$: :
MOVL	52(UCB), VCB	: 1317
CMPL	J, #1	: 1319
BNEQ	7\$: :

OA	OB	56	5C	A5	D0	00064	MOVL	92(VCB), QUOTA_CACHE	:	1327
				0F	13	00068	BEQL	7\$:	1328
				01	E5	0006A	BBCC	#1, 11(QUOTA_CACHE), 7\$:	1330
				01	DD	0006F	PUSHL	#1	:	1333
		6B		01	FB	00071	CALLS	#1, SWITCH_VOLUME	:	
05	0000G	CF		00	FB	00074	CALLS	#0, FLUSH_QUO_CACHE	:	1334
13	3A	A4		05	E0	00079	BBS	#5, 58(UCB), 8\$:	1342
	53	A5		01	E1	0007E	BBBC	#1, 83(VCB), 9\$:	1343
				53	DD	00083	PUSHL	J	:	1346
		6B		01	FB	00085	CALLS	#1, SWITCH_VOLUME	:	
				7E	D4	00088	CLRL	-(SP)	:	1347
	0000G	CF		01	FB	0008A	CALLS	#1, WRITE_DIRTY	:	
				54	DD	0008F	PUSHL	UCB	:	1348
B2	0000G	CF		01	FB	00091	CALLS	#1, KILL_CACHE	:	
		53		57	F3	00096	AOBLEQ	R7, J, 5\$:	1303
				7E	D4	0009A	CLRL	-(SP)	:	1358
08	0000G	CF		01	FB	0009C	CALLS	#1, WRITE_DIRTY	:	
		6A		04	E5	000A1	BBCC	#4, (BASE), 10\$:	1363
			08	AA	DD	000A5	PUSHL	8(BASE)	:	1364
	0000V	CF		01	FB	000A8	CALLS	#1, ZERO_WINDOWS	:	
		53	00D0	CA	D0	000AD	MOVL	208(BASE), FCB	:	1373
				50	13	000B2	BEQL	14\$:	
			18	A3	B5	000B4	TSTW	24(FCB)	:	1376
				1F	12	000B7	BNEQ	11\$:	
	08	AA		53	D1	000B9	CMPL	FCB, 8(BASE)	:	1379
				37	13	000BD	BEQL	13\$:	
	50			53	D0	000BF	MOVL	FCB, R0	:	1381
			0000V	30	000C2	BSBW	SET_DIRINDX		:	
	2E			50	E8	000C5	BLBS	R0, 13\$:	
				53	DD	000C8	PUSHL	FCB	:	1384
	0000V	CF		01	FB	000CA	CALLS	#1, DEL_EXTFCB	:	
				53	DD	000CF	PUSHL	FCB	:	1385
	0000V	CF		01	FB	000C1	CALLS	#1, NUKE_HEAD_FCB	:	
				1E	11	000D6	BRB	13\$:	1376
			1C	A3	B5	000D8	TSTW	28(FCB)	:	1392
				19	13	000DB	BEQL	13\$:	
	7E		28	A3	3C	000DD	MOVZWL	40(FCB), -(SP)	:	1395
	6B			01	FB	000E1	CALLS	#1, SWITCH_VOLUME	:	
	0A			59	E8	000E4	BLBS	CLUSTER, 12\$:	1396
			4C	A3	DD	000E7	PUSHL	76(FCB)	:	1398
				01	DD	000EA	PUSHL	#1	:	
	0000G	CF		02	FB	000EC	CALLS	#2, KILL_BUFFERS	:	
	0000V	CF		00	FB	000F1	CALLS	#0, ZERO_IDX	:	1399
		6A	00C00020	8F	CA	000F6	BICL2	#1258294\$, (BASE)	:	1411
			00D0	CA	D4	000FD	CLRL	208(BASE)	:	1412
			08	A8	D4	00101	CLRL	8(R8)	:	1413
	53		08	AA	D0	00104	MOVL	8(BASE), FCB	:	1417
				2D	13	00108	BEQL	16\$:	
0A		6A		0E	E1	0010A	BBBC	#14, (BASE), 15\$:	1426
		07		59	E9	0010E	BLBC	CLUSTER, 15\$:	1427
				53	DD	00111	PUSHL	FCB	:	1429
	0000G	CF		01	FB	00113	CALLS	#1, MAKE_FCB_STALE	:	
			18	A3	B5	00118	TSTW	24(FCB)	:	1435
				1A	12	0011B	BNEQ	16\$:	
	50			53	D0	0011D	MOVL	FCB, R0	:	1437
			0000V	30	00120	BSBW	SET_DIRINDX		:	
	11			50	E8	00123	BLBS	R0, 16\$:	

CLENUP
V04-000

E 12
16-Sep-1984 00:02:25
14-Sep-1984 12:30:12

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[F11X.SRC]CLENUP.B32;1
Page 11
(2)

0000V	CF		53	DD	00126	PUSHL	FCB	:	1441
			01	FB	00128	CALLS	#1, DEL_EXTFCB	:	
0000V	CF		53	DD	0012D	PUSHL	FCB	:	1443
			01	FB	0012F	CALLS	#1, NUKE_HEAD_FCB	:	
		08	AA	D4	00134	CLRL	8(BASE)	:	1445
	50		01	D0	00137	MOVL	#1, R0	:	1449
			04	0013A	16\$:	RET		:	1451

; Routine Size: 315 bytes, Routine Base: \$CODE\$ + 0000

CL
VO

```

464 1452 1 GLOBAL ROUTINE ZERO_WINDOWS (FCB) : L_NORM =
465 1453 1
466 1454 1 !++
467 1455 1
468 1456 1 FUNCTIONAL DESCRIPTION:
469 1457 1
470 1458 1 This routine invalidates all windows currently in use on the
471 1459 1 indicated FCB. This routine must be executed in kernel mode.
472 1460 1
473 1461 1 CALLING SEQUENCE:
474 1462 1 ZERO_WINDOWS (ARG1)
475 1463 1
476 1464 1 INPUT PARAMETERS:
477 1465 1 ARG1: address of FCB
478 1466 1
479 1467 1 IMPLICIT INPUTS:
480 1468 1 CURRENT_WINDOW: address of caller's window, if any
481 1469 1
482 1470 1 OUTPUT PARAMETERS:
483 1471 1 NONE
484 1472 1
485 1473 1 IMPLICIT OUTPUTS:
486 1474 1 NONE
487 1475 1
488 1476 1 ROUTINE VALUE:
489 1477 1 NONE
490 1478 1
491 1479 1 SIDE EFFECTS:
492 1480 1 all windows marked empty, caller's turned
493 1481 1
494 1482 1 !--
495 1483 1
496 1484 2 BEGIN
497 1485 2
498 1486 2 MAP
499 1487 2 FCB : REF BBLOCK;
500 1488 2
501 1489 2 LOCAL
502 1490 2 P : REF BBLOCK, ! window pointer
503 1491 2 DUMMY, ! dummy storage for REMQUE return
504 1492 2 WINDOW_SEGMENT : REF BBLOCK, ! pointer to window segment
505 1493 2 NEXT_SEGMENT : REF BBLOCK; ! pointer to window after next one
506 1494 2
507 1495 2 BASE_REGISTER;
508 1496 2
509 1497 2 EXTERNAL ROUTINE
510 1498 2 DEALLOCATE : L_NORM; ! deallocate dynamic memory
511 1499 2
512 1500 2 ! Loop through the window list off the FCB, zeroing all the retrieval pointer
513 1501 2 ! counts. Then turn the user's window to VBN 1 if it exists.
514 1502 2 !
515 1503 2
516 1504 2 P = .FCB[FCB$L_WLFL];
517 1505 2
518 1506 2 UNTIL .P EQL FCB[FCB$L_WLFL] DO
519 1507 2 BEGIN
520 1508 2 P[WCBSW_NMAP] = 0;

```



```

521 1509 3 WINDOW_SEGMENT = .P[WCBSL_LINK];
522 1510 3 UNTIL .WINDOW_SEGMENT EQL 0
523 1511 3 DO
524 1512 4 BEGIN
525 1513 4 NEXT_SEGMENT = .WINDOW_SEGMENT[WCBSL_LINK];
526 1514 4 REMQUE (.WINDOW_SEGMENT, DUMMY);
527 1515 4 DEALLOCATE (.WINDOW_SEGMENT);
528 1516 4 WINDOW_SEGMENT = .NEXT_SEGMENT;
529 1517 4 END;
530 1518 3 P[WCBSL_LINK] = 0;
531 1519 3 P[WCBSV_COMPLETE] = 0;
532 1520 3 P = .P[WCBSL_WLFL];
533 1521 3 END;
534 1522 2
535 1523 2 ! ***** Note: When handling of window misses goes into its final form,
536 1524 2 ! this routine must also scan the I/O queue on the UCB and look for I/O
537 1525 2 ! into the blocks just deallocated. All such requests must be yanked out
538 1526 2 ! of the queue and routed to the ACP for error processing.
539 1527 2
540 1528 2 RETURN 1;
541 1529 2
542 1530 1 END;
! end of routine ZERO_WINDOWS
```

				.EXTRN	DEALLOCATE	
			003C 00000	.ENTRY	ZERO_WINDOWS, Save R2,R3,R4,R5	: 1452
	50	04	AC D0 00002	MOVL	FCB, R0	: 1504
	52	10	A0 D0 00006	MOVL	16(R0), P	
50	04	AC	10 C1 0000A 1\$:	ADDL3	#16, FCB, R0	: 1506
	50		52 D1 0000F	CMPL	P, R0	
			28 13 00012	BEQL	4\$	
		16	A2 B4 00014	CLRW	22(P)	: 1508
	53	20	A2 D0 00017	MOVL	32(P), WINDOW_SEGMENT	: 1509
			13 13 0001B 2\$:	BEQL	3\$: 1510
	54	20	A3 D0 0001D	MOVL	32(WINDOW_SEGMENT), NEXT_SEGMENT	: 1513
	55		63 0F 00021	REMQUE	(WINDOW_SEGMENT), DUMMY	: 1514
			53 DD 00024	PUSHL	WINDOW_SEGMENT	: 1515
0000G	CF		01 FB 00026	CALLS	#1, DEALLOCATE	
	53		54 D0 0002B	MOVL	NEXT_SEGMENT, WINDOW_SEGMENT	: 1516
			EB 11 0002E	BRB	2\$: 1510
		20	A2 D4 00030 3\$:	CLRL	32(P)	: 1518
	0B	A2	20 8A 00033	BICB2	#32, 11(P)	: 1519
	52		62 D0 00037	MOVL	(P), P	: 1520
			CE 11 0003A	BRB	1\$: 1506
	50		01 D0 0003C 4\$:	MOVL	#1, R0	: 1528
			04 0003F	RET		: 1530

; Routine Size: 64 bytes, Routine Base: \$CODE\$ + 013B

```

: 544 1531 1 GLOBAL ROUTINE ZERO_IDX : L_NORM NOVALUE =
: 545 1532 1
: 546 1533 1 !++
: 547 1534 1
: 548 1535 1 FUNCTIONAL DESCRIPTION:
: 549 1536 1
: 550 1537 1 This routine initializes the index in a directory FCB to an unknown
: 551 1538 1 state. It will be rebuilt with the next several lookups.
: 552 1539 1 It also bumps the sequence count to indicate a change in contents.
: 553 1540 1
: 554 1541 1
: 555 1542 1 CALLING SEQUENCE:
: 556 1543 1 ZERO_IDX ()
: 557 1544 1
: 558 1545 1 INPUT PARAMETERS:
: 559 1546 1 NONE
: 560 1547 1
: 561 1548 1 IMPLICIT INPUTS:
: 562 1549 1 DIR_FCB: directory FCB to init
: 563 1550 1
: 564 1551 1 OUTPUT PARAMETERS:
: 565 1552 1 NONE
: 566 1553 1
: 567 1554 1 IMPLICIT OUTPUTS:
: 568 1555 1 NONE
: 569 1556 1
: 570 1557 1 ROUTINE VALUE:
: 571 1558 1 1
: 572 1559 1
: 573 1560 1 SIDE EFFECTS:
: 574 1561 1 directory index zeroed
: 575 1562 1
: 576 1563 1 !--
: 577 1564 1
: 578 1565 2 BEGIN
: 579 1566 2
: 580 1567 2 BIND_COMMON;
: 581 1568 2
: 582 1569 2 LOCAL
: 583 1570 2 DIRINDX : REF BBLOCK FIELD (DIRC);
: 584 1571 2
: 585 1572 2 DIR_FCB[FCB$W_DIRSEQ] = .DIR_FCB[FCB$W_DIRSEQ] + 1;
: 586 1573 2
: 587 1574 2 IF (DIRINDX = .DIR_FCB [FCB$L_DIRINDX]) NEQ 0
: 588 1575 2 THEN
: 589 1576 2 DIRINDX [DIRC$W_INUSE] = 0;
: 590 1577 2
: 591 1578 1 END;

```

! end of routine ZERO_IDX

			0000 00000	.ENTRY	ZERO_IDX, Save nothing	: 1531
50	00D0	CA	D0 00002	MOVL	208(BASE), R0	: 1572
	42	AO	B6 00007	INCL	66(R0)	
50	00D0	CA	D0 0000A	MOVL	208(BASE), R0	: 1574

CLENUP
V04-000

I 12
16-Sep-1984 00:02:25
14-Sep-1984 12:30:12

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[F11X.SRC]CLENUP.B32;1 Page 15
(4)

50	00B0	C0	D0	0000F	MOVL	176(R0), DIRINDX	:
		02	13	00014	BEQL	1\$:
		60	B4	00016	CLRW	(DIRINDX)	:
		04	00018	1\$:	RET		: 1576
							: 1578

; Routine Size: 25 bytes, Routine Base: \$CODE\$ + 017B

```

593 1579 1 GLOBAL ROUTINE ERR_CLEANUP : L_NORM =
594 1580 1
595 1581 1 ++
596 1582 1
597 1583 1 FUNCTIONAL DESCRIPTION:
598 1584 1
599 1585 1 This routine performs the cleanup needed after a file
600 1586 1 operation that has terminated in an error.
601 1587 1
602 1588 1 CALLING SEQUENCE:
603 1589 1 ERR_CLEANUP ()
604 1590 1
605 1591 1 INPUT PARAMETERS:
606 1592 1 NONE
607 1593 1
608 1594 1 IMPLICIT INPUTS:
609 1595 1 CLEANUP_FLAGS: indicate specific actions to do
610 1596 1
611 1597 1 OUTPUT PARAMETERS:
612 1598 1 NONE
613 1599 1
614 1600 1 IMPLICIT OUTPUTS:
615 1601 1 NONE
616 1602 1
617 1603 1 ROUTINE VALUE:
618 1604 1 NONE
619 1605 1
620 1606 1 SIDE EFFECTS:
621 1607 1 file deaccessed if necessary
622 1608 1 channel window pointer cleared
623 1609 1
624 1610 1 --
625 1611 1
626 1612 2 BEGIN
627 1613 2
628 1614 2 BIND_COMMON;
629 1615 2
630 1616 2 DIR_CONTEXT_DEF;
631 1617 2
632 1618 2 EXTERNAL ROUTINE
633 1619 2 REBLD_PRIM_FCB : L_NORM NOVALUE, ! rebuild primary fcb from header
634 1620 2 BUILD_EXT_FCBS : L_NORM NOVALUE, ! build extension fcb chain
635 1621 2 ALLOCATION_UNLOCK : L_NORM NOVALUE, ! release allocation lock
636 1622 2 KILL_DINDX : L_NORM NOVALUE, ! release directory index block
637 1623 2 PMS_END_SUB : L_NORM, ! end metering of current subfunction
638 1624 2 CLOSE_FILE : L_NORM, ! close internal file
639 1625 2 DEACC_QFILE : L_NORM, ! deaccess the quota file
640 1626 2 DEALLOCATE : L_NORM, ! deallocate dynamic memory
641 1627 2 SEND_SYMBIONT : L_NORM ADDRESSING_MODE (GENERAL), ! send file to job controller
642 1628 2
643 1629 2 SWITCH_VOLUME : L_NORM, ! switch to desired volume
644 1630 2 RESTORE_DIR : L_NORM, ! restore directory context
645 1631 2 DIR_SCAN : L_NORM, ! scan directory file
646 1632 2 MAKE_ENTRY : L_NORM, ! create new directory entry
647 1633 2 REMOVE : L_NORM, ! remove a directory entry
648 1634 2 READ_BLOCK : L_NORM, ! read a disk block
649 1635 2 MARK_DIRTY : L_NORM, ! mark disk block for write back

```



```

: 650      1636 2      WRITE_BLOCK      : L_NORM,      ! write a disk block
: 651      1637 ~      DELETE_FILE      : L_NORM,      ! delete a file
: 652      1638 ~      DELETE_FID      : L_NORM,      ! delete a file number
: 653      1639 ~      RETURN_BLOCKS    : L_NORM,      ! return blocks to storage map
: 654      1640 ~      TRUNCATE         : L_NORM,      ! file truncate routine
: 655      1641 ~      INVALIDATE      : L_NORM,      ! invalidate a buffer
: 656      1642 ~      READ_HEADER      : L_NORM,      ! read file header
: 657      1643 ~      CHECKSUM         : L_NORM,      ! checksum file header
: 658      1644 ~      REMAP_FILE       : L_NORM;      ! rebuild the windows for a file
: 659      1645 ~
: 660      1646 ~
: 661      1647 ~      ! If a subfunction was being executed, turn off metering now.
: 662      1648 ~
: 663      1649 ~
: 664      1650 ~      IF .PMS_SUB_NEST NEQ 0
: 665      1651 ~      THEN
: 666      1652 ~          BEGIN
: 667      1653 ~              PMS_SUB_NEST = 1;
: 668      1654 ~              PMS_END_SUB ();
: 669      1655 ~          END;
: 670      1656 ~
: 671      1657 ~      ! We repeat the entire procedure twice if a secondary file operation was
: 672      1658 ~      ! in progress (indicated by non-zero saved context).
: 673      1659 ~
: 674      1660 ~
: 675      1661 ~      WHILE 1 DO
: 676      1662 ~      BEGIN
: 677      1663 ~
: 678      1664 ~      ! Locals are declared here to prevent their scope from extending around the
: 679      1665 ~      ! entire main loop and raising havoc with register assignment.
: 680      1666 ~
: 681      1667 ~
: 682      1668 ~      LOCAL
: 683      1669 ~          NAME DESC      : BBLOCK [FND_LENGTH], ! file name descriptor block
: 684      1670 ~          HEADER      : REF BBLOCK,      ! address of file header
: 685      1671 ~          IDENT_AREA   : REF BBLOCK,      ! ident area of file header
: 686      1672 ~          FCB         : REF BBLOCK,      ! FCB pointer
: 687      1673 ~          WINDOW_SEGMENT : REF BBLOCK,      ! address of the next window segment
: 688      1674 ~          NEXT_SEGMENT  : REF BBLOCK,      ! address of one beyond the next window
: 689      1675 ~          RECAHDR      : REF BBLOCK,      ! address of directory record
: 690      1676 ~          DIR_FLAGS     : BITVECTOR [32], ! directory cleanup flags
: 691      1677 ~          UNREC_LOCAL,    : local copy of UNREC COUNT
: 692      1678 ~          FID_LOCAL,      : local copy of NEW_FID
: 693      1679 ~          T1,              : random temps
: 694      1680 ~          T2,
: 695      1681 ~          T3;
: 696      1682 ~
: 697      1683 ~      ! Show that cleanup is in progress.
: 698      1684 ~
: 699      1685 ~
: 700      1686 ~      CLEANUP_FLAGS[CLF_CLEANUP] = 1;
: 701      1687 ~
: 702      1688 ~      ! If the ref count on the primary fcb was biased in fid_to_spec, remove
: 703      1689 ~      ! the bias.
: 704      1690 ~
: 705      1691 ~
: 706      1692 ~      IF TESTBITSC (CLEANUP_FLAGS [CLF_PFCB_REF_UP])

```

```

707 1693 3 THEN
708 1694     PRIMARY_FCB [FCB$W_REFCNT] = .PRIMARY_FCB [FCB$W_REFCNT] - 1;
709 1695
710 1696     ! If an internal file is open, close it first.
711 1697     !
712 1698
713 1699     IF TESTBITSC (CLEANUP_FLAGS[CLF_CLOSEFILE])
714 1700     THEN CLOSE_FILE (.CURRENT_WINDOW);
715 1701
716 1702     ! Invalidate the file ID cache, if necessary.
717 1703     !
718 1704
719 1705     IF TESTBITSC (CLEANUP_FLAGS[CLF_FLUSHFID])
720 1706     THEN KERNEL_CALL (FLUSH_FIDCACHE);
721 1707
722 1708     ! Deaccess the quota file, if we were in the final stages of a quota file
723 1709     ! enable.
724 1710     !
725 1711
726 1712     IF TESTBITSC (CLEANUP_FLAGS[CLF_DEACCQFILE])
727 1713     THEN KERNEL_CALL (DEACC_QFILE);
728 1714
729 1715     ! If there is a file header resident, it probably needs to be checksummed.
730 1716     !
731 1717
732 1718     IF .FILE_HEADER NEQ 0
733 1719     THEN CHECKSUM (.FILE_HEADER);
734 1720
735 1721     ! Clean out the window pointer in the user's channel if necessary.
736 1722     !
737 1723
738 1724     IF TESTBITSC (CLEANUP_FLAGS[CLF_ZCHANNEL])
739 1725     THEN KERNEL_CALL (ZERO_CHANNEL);
740 1726
741 1727     ! If there are unrecorded blocks allocated from the storage map, return them.
742 1728     !
743 1729
744 1730     IF (UNREC_LOCAL = .UNREC_COUNT) NEQ 0
745 1731     THEN
746 1732         BEGIN
747 1733             UNREC_COUNT = 0;
748 1734             SWITCH_VOLUME (.UNREC_RVN);
749 1735             RETURN_BLOCKS (.UNREC_LBN, .UNREC_LOCAL, DO_NOT_ERASE);
750 1736         END;
751 1737
752 1738     ! If there is a dangling file ID (from a partial create or header extension),
753 1739     ! dispose of it.
754 1740     !
755 1741
756 1742     IF (FID_LOCAL = .NEW_FID) NEQ 0
757 1743     THEN
758 1744         BEGIN
759 1745             NEW_FID = 0;
760 1746             SWITCH_VOLUME (.NEW_FID_RVN);
761 1747             DELETE_FID (.FID_LOCAL);
762 1748         END;
763 1749

```



```

764 1750 3 ! Get back the primary file header of the file in process.
765 1751 3 !
766 1752 3 !
767 1753 3 HEADER = 0;
768 1754 3 IF .FILE_HEADER NEQ 0
769 1755 3 THEN
770 1756 4 BEGIN
771 1757 4 FILE_HEADER = 0;
772 1758 4 IF (CURR_LCKINDX = .PRIM_LCKINDX) NEQ 0
773 1759 4 THEN
774 1760 5 HEADER = READ_HEADER ((IF .CURRENT_FIB NEQ 0
775 1761 5 THEN CURRENT_FIB[FIB$W_FID]
776 1762 4 ELSE 0),
777 1763 4 .PRIMARY_FCB);
778 1764 3 END;
779 1765 3 !
780 1766 3 ! Send the file to the job controller if it is to be spooled.
781 1767 3 !
782 1768 3 !
783 1769 3 IF TESTBITSC (CLEANUP_FLAGS[CLF_DOSPOOL])
784 1770 3 THEN
785 1771 4 BEGIN
786 1772 4 !
787 1773 4 ! Make sure the allocation lock is released before sending it
788 1774 4 ! to the symbiont to avoid potential deadlock with the symbiont.
789 1775 4 !
790 1776 4 !
791 1777 4 ALLOCATION_UNLOCK ();
792 1778 4 SEND_SYMBIONT (.HEADER, .PRIMARY_FCB);
793 1779 3 END;
794 1780 3 !
795 1781 3 ! Deaccess the file if requested.
796 1782 3 !
797 1783 3 !
798 1784 3 IF TESTBITSC (CLEANUP_FLAGS[CLF_DEACCESS])
799 1785 3 THEN KERNEL_CALL (MAKE_DEACCESS);
800 1786 3 !
801 1787 3 ! Deallocate the window block if called for.
802 1788 3 !
803 1789 3 !
804 1790 3 IF TESTBITSC (CLEANUP_FLAGS[CLF_DELWINDOW])
805 1791 3 THEN
806 1792 3 IF .CURRENT_WINDOW NEQ 0
807 1793 3 THEN
808 1794 4 BEGIN
809 1795 4 WINDOW_SEGMENT = .CURRENT_WINDOW;
810 1796 4 DO
811 1797 5 BEGIN
812 1798 5 NEXT_SEGMENT = .WINDOW_SEGMENT[WCBSL_LINK];
813 1799 5 KERNEL_CALL (DEALLOCATE, .WINDOW_SEGMENT);
814 1800 5 WINDOW_SEGMENT = .NEXT_SEGMENT;
815 1801 5 END
816 1802 4 UNTIL .WINDOW_SEGMENT EQL 0;
817 1803 4 CURRENT_WINDOW = 0;
818 1804 3 END;
819 1805 3 !
820 1806 3 ! Fix the file header back link, if it was modified.
```

```

821 1807 3 !
822 1808 3
823 1809 3 IF TESTBITSC (CLEANUP_FLAGS[CLF_FIXLINK])
824 1810 3 THEN IF .HEADER NEQ 0
825 1811 3 THEN
826 1812 4 BEGIN
827 1813 4 CH$MOVE (FID$ LENGTH, PREV LINK, HEADER[FH2$W BACKLINK]);
828 1814 4 IDENT AREA = .HEADER + .HEADER[FH2$B IDOFFSET]*2;
829 1815 4 CH$MOVE (MINU (FILENAME LENGTH, FI2$$_FILENAME), PREV_INAME,
830 1816 4 IDENT AREA[F12$T FILENAME]);
831 1817 4 CH$MOVE (MINU (FILENAME LENGTH-FI2$$_FILENAME, FI2$$_FILENAMEEXT),
832 1818 4 PREV_INAME+FI2$$_FILENAME,
833 1819 4 IDENT AREA[F12$T FILENAMEEXT]);
834 1820 4 CHECKSUM (.HEADER);
835 1821 4 MARK_DIRTY (.HEADER);
836 1822 3 END;
837 1823 3
838 1824 3 ! If a file deletion is called for, do it. This is either a create that
839 1825 3 ! failed later on, or a real delete.
840 1826 3 !
841 1827 3
842 1828 3 IF TESTBITSC (CLEANUP_FLAGS[CLF_DELFILE])
843 1829 3 THEN IF .HEADER NEQ 0
844 1830 3 THEN
845 1831 4 BEGIN
846 1832 4 IF .PRIMARY_FCB NEQ 0
847 1833 4 THEN
848 1834 4 IF .PRIMARY_FCB [FCB$L_DIRINDX] NEQ 0
849 1835 4 THEN
850 1836 4 KILL_DINDX (.PRIMARY_FCB);
851 1837 4
852 1838 4 CLEANUP_FLAGS[CLF_TRUNCATE] = 0; ! no truncate necessary after a delete
853 1839 4 DELETE_FILE (.CURRENT_FIB, .HEADER);
854 1840 3 END;
855 1841 3
856 1842 3 ! If an extend operation failed, truncate the file.
857 1843 3 !
858 1844 3
859 1845 3 IF TESTBITSC (CLEANUP_FLAGS[CLF_TRUNCATE])
860 1846 3 THEN IF .HEADER NEQ 0
861 1847 3 THEN
862 1848 4 BEGIN
863 1849 4 T1 = .CURRENT_FIB[FIB$L_EXSZ]; ! save the data returned by EXTEND
864 1850 4 T2 = .CURRENT_FIB[FIB$L_EXVBN]; ! so it won't be smashed by TRUNCATE
865 1851 4 T3 = .USER_STATUS[1];
866 1852 4 CURRENT_FIB[FIB$L_EXSZ] = 0;
867 1853 4 TRUNCATE (.CURRENT_FIB, .HEADER, .T2);
868 1854 4 HEADER = .FILE HEADER; ! follow buffer shuffling
869 1855 4 CURRENT_FIB[FIB$L_EXSZ] = .T1;
870 1856 4 CURRENT_FIB[FIB$L_EXVBN] = .T2;
871 1857 4 USER_STATUS[1] = .T3;
872 1858 4 CLEANUP_FLAGS[CLF_INVWINDOW] = 0; ! windows were never extended, so no need
873 1859 4 CHECKSUM (.HEADER);
874 1860 3 END;
875 1861 3
876 1862 3 ! Various errors leave the file control block screwed up. If needed,
877 1863 3 ! rebuild it and its extensions from scratch.
```



```

878      1864      3      !
879      1865      3      !
880      1866      3      IF TESTBITSC (CLEANUP_FLAGS[CLF_FIXFCB])
881      1867      3      AND .HEADER NEQ 0
882      1868      3      THEN
883      1869      4      BEGIN
884      1870      4
885      1871      4      REBLD_PRIM_FCB (.PRIMARY_FCB, .HEADER);
886      1872      4
887      1873      4      BUILD_EXT_FCBS (.HEADER);
888      1874      4
889      1875      3      END;
890      1876      3
891      1877      3      ! Cleanup any cathedral windows which have broken.
892      1878      3      !
893      1879      3
894      1880      3      IF TESTBITSC (CLEANUP_FLAGS[CLF_REMAP]) THEN REMAP_FILE ();
895      1881      3
896      1882      3      ! Do directory operation cleanups. We could have entered a new file, removed
897      1883      3      ! an old one, or both, or done a supersede. A supersede is a replacement of
898      1884      3      ! the FID for the same name, type, and version.
899      1885      3      !
900      1886      3
901      1887      3      DIR_FLAGS = .CLEANUP_FLAGS;
902      1888      3      CLEANUP_FLAGS[CLF_SUPERSEDE] = 0;
903      1889      3      CLEANUP_FLAGS[CLF_REENTER] = 0;
904      1890      3      CLEANUP_FLAGS[CLF_REMOVE] = 0;
905      1891      3
906      1892      3      IF .DIR_FLAGS[CLF_SUPERSEDE]
907      1893      3      OR .DIR_FLAGS[CLF_REENTER]
908      1894      3      OR .DIR_FLAGS[CLF_REMOVE]
909      1895      3      THEN
910      1896      4      BEGIN
911      1897      4      SWITCH_VOLUME (.CURRENT_FIB[FIB$W_DID_RVN]);
912      1898      4
913      1899      4      ! Buffer pool thrashing may have kicked out the directory block we need.
914      1900      4      ! re-read it and recompute the buffer pointers.
915      1901      4      !
916      1902      4
917      1903      4      IF .DIR_ENTRY NEQ 0
918      1904      4      THEN RESTORE_DIR (DIR_CONTEXT);
919      1905      4
920      1906      4      ! If a directory entry needs to be removed, do so. Pointers are all set
921      1907      4      ! up for the REMOVE routine.
922      1908      4      !
923      1909      4
924      1910      4      IF .DIR_FLAGS[CLF_REMOVE]
925      1911      4      THEN REMOVE (0);
926      1912      4
927      1913      4      ! If a directory entry needs to be re-entered, do so. If the entry was
928      1914      4      ! removed through an auto-purge, we need to rescan to the point of
929      1915      4      ! removal because a directory shuffle may have invalidated the
930      1916      4      ! pointers. Construct a name descriptor from the saved name and version
931      1917      4      ! and call the enter routine.
932      1918      4      !
933      1919      4
934      1920      4      IF .DIR_FLAGS[CLF_REENTER]
```



```

: 935      1921  4      THEN
: 936      1922  5      BEGIN
: 937      1923  5      CH$FILL (0, FND_LENGTH, NAME_DESC);
: 938      1924  5      NAME_DESC[FND_COUNT] = .PREV_NAME[0];
: 939      1925  5      NAME_DESC[FND_STRING] = PREV_NAME[1];
: 940      1926  5      NAME_DESC[FND_VERSION] = .PREV_VERSION;
: 941      1927  5      IF .DIR_FLAGS[CLF_SUPERSEDE]
: 942      1928  5      THEN
: 943      1929  6      BEGIN
: 944      1930  6      LAST_ENTRY[0] = 0;
: 945      1931  6      DIR_SCAN (NAME_DESC, 0, 0, 0, 0, 0, -1);
: 946      1932  6      CH$MOVE (FID$C_LENGTH, SUPER_FID, CURRENT_FIB[FIB$W_FID]);
: 947      1933  5      END;
: 948      1934  5      MAKE_ENTRY (NAME_DESC, .CURRENT_FIB);
: 949      1935  5      CLEANUP_FLAGS[CLF_REMOVE] = 0;
: 950      1936  5      WRITE_BLOCK (.DIR_BUFFER);
: 951      1937  4      END;
: 952      1938  4
: 953      1939  4      ! A supersede cleanup consists simply of replacing the superseded file ID
: 954      1940  4      ! in the directory record. Note that the supersede bit could also be set
: 955      1941  4      ! by a create/auto-purge, which also sets the remove and enter bits, and
: 956      1942  4      ! is handled above.
: 957      1943  4      !
: 958      1944  4
: 959      1945  4      IF .DIR_FLAGS[CLF_SUPERSEDE]
: 960      1946  4      AND NOT .DIR_FLAGS[CLF_REENTER]
: 961      1947  4      AND NOT .DIR_FLAGS[CLF_REMOVE]
: 962      1948  4      THEN
: 963      1949  5      BEGIN
: 964      1950  5      DIR_VERSION[DIR$W_VERSION] = .PREV_VERSION;
: 965      1951  5      CH$MOVE (FIB$S_FID, SUPER_FID, DIR_VERSION[DIR$W_FID]);
: 966      1952  5      MARK_DIRTY (.DIR_BUFFER);
: 967      1953  5      END
: 968      1954  5
: 969      1955  5
: 970      1956  3      END;                                ! end of directory cleanup processing
: 971      1957  3
: 972      1958  3      ! Copy the saved context, if any back into the primary context and repeat
: 973      1959  3      ! the cleanup.
: 974      1960  3      !
: 975      1961  3
: 976      1962  3      IF .CONTEXT_SAVE EQL 0 THEN EXITLOOP;
: 977      1963  3      CH$MOVE (CONTEXT_SIZE, CONTEXT_SAVE, CONTEXT_START);
: 978      1964  3      CONTEXT_SAVE = 0;
: 979      1965  3
: 980      1966  2      END;                                ! end of major loop
: 981      1967  2
: 982      1968  2      RETURN 1;
: 983      1969  2
: 984      1970  1      END;                                ! end of routine ERR_CLEANUP
```

```

      .EXTRN REBLD_PRIM_FCB, BUILD_EXT_FCBS
      .EXTRN ALLOCATION_UNLOCK
      .EXTRN KILL_DINDX, PMS_END_SUB
      .EXTRN CLOSE_FILE, DEACC_QFILE
```


			04	AA	D4	000A8	CLRL	4(BASE)	: 1757
	14	AA	18	AA	D0	000AB	MOVL	24(BASE), 20(BASE)	: 1758
				19	13	000B0	BEQL	12\$:
			00	BE	DD	000B2	PUSHL	@0(SP)	: 1763
				69	D5	000B5	TSTL	(R9)	: 1760
				08	13	000B7	BEQL	10\$:
50		69		04	C1	000B9	ADDL3	#4, (R9), R0	: 1761
				50	DD	000BD	PUSHL	R0	:
				02	11	000BF	BRB	11\$:
				7E	D4	000C1	CLRL	-(SP)	:
	0000G	CF		02	FB	000C3	CALLS	#2, READ HEADER	: 1760
		56		50	D0	000C8	MOVL	R0, HEADER	:
11		6A		02	E5	000CB	BBCC	#2, (BASE), 13\$: 1769
	0000G	CF		00	FB	000CF	CALLS	#0, ALLOCATION_UNLOCK	: 1777
			00	BE	DD	000D4	PUSHL	@0(SP)	: 1778
				56	DD	000D7	PUSHL	HEADER	:
	00000000G	00		02	FB	000D9	CALLS	#2, SEND SYMBIONT	:
05		6A		10	E5	000E0	BBCC	#16, (BASE), 14\$: 1784
	0000V	CF		00	FB	000E4	CALLS	#0, MAKE DEACCESS	: 1785
1C		6A		1A	E5	000E9	BBCC	#26, (BASE), 16\$: 1790
			0C	AA	D5	000ED	TSTL	12(BASE)	: 1792
				17	13	000F0	BEQL	16\$:
		52	0C	AA	D0	000F2	MOVL	12(BASE), WINDOW_SEGMENT	: 1795
		53	20	A2	D0	000F6	MOVL	32(WINDOW_SEGMENT), NEXT_SEGMENT	: 1798
				52	DD	000FA	PUSHL	WINDOW_SEGMENT	: 1799
	0000G	CF		01	FB	000FC	CALLS	#1, DEALLOCATE	:
		52		53	D0	00101	MOVL	NEXT_SEGMENT, WINDOW_SEGMENT	: 1800
				F0	12	00104	BNEQ	15\$: 1802
			0C	AA	D4	00106	CLRL	12(BASE)	: 1803
29		6A		1E	E5	00109	BBCC	#30, (BASE), 17\$: 1809
				56	D5	0010D	TSTL	HEADER	: 1810
				25	13	0010F	BEQL	17\$:
42	A6	30	AA	06	28	00111	MOVC3	#6, 48(BASE), 66(HEADER)	: 1813
			50	66	9A	00117	MOVZBL	(HEADER), R0	: 1814
			58	6640	3E	0011A	MOVAV	(HEADER)[R0], IDENT_AREA	:
	68		6B	14	28	0011E	MOVC3	#20, (R11), (IDENT_AREA)	: 1816
36	A8	14	AB	3C	28	00122	MOVC3	#60, 20(R11), 54(IDENT_AREA)	: 1819
				56	DD	00128	PUSHL	HEADER	: 1820
	0000G	CF		01	FB	0012A	CALLS	#1, CHECKSUM	:
				56	DD	0012F	PUSHL	HEADER	: 1821
	0000G	CF		01	FB	00131	CALLS	#1, MARK DIRTY	:
24		6A		15	E5	00136	BBCC	#21, (BASE), 19\$: 1828
				56	D5	0013A	TSTL	HEADER	: 1829
				20	13	0013C	BEQL	19\$:
		50	00	BE	D0	0013E	MOVL	@0(SP), R0	: 1832
				0D	13	00142	BEQL	18\$:
			00B0	C0	D5	00144	TSTL	176(R0)	: 1834
				07	13	00148	BEQL	18\$:
				50	DD	0014A	PUSHL	R0	: 1836
	0000G	CF		01	FB	0014C	CALLS	#1, KILL DINDX	:
	02	AA		04	8A	00151	BICB2	#4, 2(BASE)	: 1838
				56	DD	00155	PUSHL	HEADER	: 1839
				69	DD	00157	PUSHL	(R9)	:
	0000G	CF		02	FB	00159	CALLS	#2, DELETE FILE	:
4F		6A		12	E5	0015E	BBCC	#18, (BASE), 20\$: 1845
				56	D5	00162	TSTL	HEADER	: 1846
				4B	13	00164	BEQL	20\$:

50	04	50	18	69	D0	00166	MOVL	(R9), R0	1849
		54		A0	D0	00169	MOVL	24(R0), T1	
		50	1C	69	D0	0016D	MOVL	(R9), R0	1850
		52		A0	D0	00170	MOVL	28(R0), T2	
		AE		04	C1	00174	ADDL3	#4, 4(SP), R0	1851
		53		60	D0	00179	MOVL	(R0), T3	
		50	18	69	D0	0017C	MOVL	(R9), R0	1852
				A0	D4	0017F	CLRL	24(R0)	
				52	DD	00182	PUSHL	T2	1853
				56	DD	00184	PUSHL	HEADER	
	0000G	CF		69	DD	00186	PUSHL	(R9)	
		56	04	03	FB	00188	CALLS	#3, TRUNCATE	
		50		AA	D0	0018D	MOVL	4(BASE), HEADER	1854
	18	A0		69	D0	00191	MOVL	(R9), R0	1855
		50		54	D0	00194	MOVL	T1, 24(R0)	
	1C	A0		69	D0	00198	MOVL	(R9), R0	1856
50	04	AE		52	D0	0019B	MOVL	T2, 28(R0)	
		60		04	C1	0019F	ADDL3	#4, 4(SP), R0	1857
		6A		53	D0	001A4	MOVL	T3, (R0)	
				10	8A	001A7	BICB2	#16, (BASE)	1858
				56	DD	001AA	PUSHL	HEADER	1859
15	0000G	CF		01	FB	001AC	CALLS	#1, CHECKSUM	
		6A		01	E5	001B1	BBCC	#1, (BASE), 21\$	1866
				56	D5	001B5	TSTL	HEADER	1867
				11	13	001B7	BEQL	21\$	
			04	56	DD	001B9	PUSHL	HEADER	1871
				BE	DD	001BB	PUSHL	24(SP)	
	0000G	CF		02	FB	001BE	CALLS	#2, REBLD_PRIM_FCB	
		6A		56	DD	001C3	PUSHL	HEADER	1873
05	0000G	CF		01	FB	001C5	CALLS	#1, BUILD_EXT_FCBS	
		6A		1F	E5	001CA	BBCC	#31, (BASE), 22\$	1880
	0000G	CF		00	FB	001CE	CALLS	#0, REMAP_FILE	
		58		6A	D0	001D3	MOVL	(BASE), DIR_FLAGS	1887
		6A	00C00020	8F	CA	001D6	BICL2	#12582944, (BASE)	1890
08		58		05	E0	001DD	BBS	#5, DIR_FLAGS, 23\$	1892
07		58		17	E0	001E1	BBS	#23, DIR_FLAGS, 23\$	1893
03		58		16	E0	001E5	BBS	#22, DIR_FLAGS, 23\$	1894
			009E	31	001E9	BRW	28\$		
		50		69	D0	001EC	MOVL	(R9), R0	1897
		7E	0E	A0	3C	001EF	MOVZWL	14(R0), -(SP)	
	0000G	CF		01	FB	001F3	CALLS	#1, SWITCH_VOLUME	
			08	A7	D5	001F8	TSTL	8(R7)	1903
				07	13	001FB	BEQL	24\$	
				57	DD	001FD	PUSHL	R7	1904
07	0000G	CF		01	FB	001FF	CALLS	#1, RESTORE_DIR	
		58		16	E1	00204	BBCC	#22, DIR_FLAGS, 25\$	1910
				7E	D4	00208	CLRL	-(SP)	1911
	0000G	CF		01	FB	0020A	CALLS	#1, REMOVE	
52		58		17	E1	0020F	BBCC	#23, DIR_FLAGS, 27\$	1920
00		6E		00	2C	00213	MOVCS	#0, (SP), #0, #16, NAME_DESC	1923
			08	AE		00218			
	0C	AE	0156	CA	9A	0021A	MOVZBL	342(BASE), NAME_DESC+4	1924
	10	AE	0157	CA	9E	00220	MOVAB	343(BASE), NAME_DESC+8	1925
	14	AE	0152	CA	B0	00226	MOVW	338(BASE), NAME_DESC+12	1926
1E		58		05	E1	0022C	BBC	#5, DIR_FLAGS, 26\$	1927
			1C	A7	94	00230	CLRB	28(R7)	1930
		7E		01	CE	00233	MNEGL	#1, -(SP)	1931

				7E 7C 00236	CLRQ	-(SP)	:
				7E 7C 00238	CLRQ	-(SP)	:
				7E D4 0023A	CLRL	-(SP)	:
			20	AE 9F 0023C	PUSHAB	NAME DESC	:
		0000G	CF	07 FB 0023F	CALLS	#7, DIR_SCAN	:
		50		69 D0 00244	MOVL	(R9), R0	1932
04	A0	01FE	CA	06 28 00247	MOVC3	#6, 510(BASE), 4(R0)	:
				69 DD 0024E 26\$:	PUSHL	(R9)	1934
			0C	AE 9F 00250	PUSHAB	NAME DESC	:
		0000G	CF	02 FB 00253	CALLS	#2, MAKE_ENTRY	:
		02	AA	8F 8A 00258	BICB2	#64, 2(BASE)	1935
			40	A7 DD 0025D	PUSHL	4(R7)	1936
		0000G	CF	01 FB 00260	CALLS	#1, WRITE_BLOCK	:
	21		58	05 E1 00265 27\$:	BBC	#5, DIR_FLAGS, 28\$	1945
	1D		58	17 E0 00269	BBS	#23, DIR_FLAGS, 28\$	1946
	19		58	16 E0 0026D	BBS	#22, DIR_FLAGS, 28\$	1947
		0C	B7	CA B0 00271	MOVW	338(BASE), @12(R7)	1950
			50	A7 D0 00277	MOVL	12(R7), R0	1951
02	A0	01FE	CA	06 28 0027B	MOVC3	#6, 510(BASE), 2(R0)	:
			04	A7 DD 00282	PUSHL	4(R7)	1952
		0000G	CF	01 FB 00285	CALLS	#1, MARK_DIRTY	:
			36	AA D5 0028A 28\$:	TSTL	54(BASE)	1962
				0B 13 0028D	BEQL	29\$:
	6A	36	AA	36 28 0028F	MOVC3	#54, 54(BASE), (BASE)	1963
			36	AA D4 00294	CLRL	54(BASE)	1964
				FD8F 31 00297	BRW	1\$	1661
		50		01 D0 0029A 29\$:	MOVL	#1, R0	1968
				04 0029D	RET		1970

; Routine Size: 670 bytes, Routine Base: \$CODE\$ + 0194


```

: 986 1971 1 ROUTINE FLUSH_FIDCACHE : L_NORM =
: 987 1972 1
: 988 1973 1 ++
: 989 1974 1
: 990 1975 1 FUNCTIONAL DESCRIPTION:
: 991 1976 1
: 992 1977 1 This routine empties the file ID cache by zeroing the entry count.
: 993 1978 1 It must be called in kernel mode.
: 994 1979 1
: 995 1980 1
: 996 1981 1 CALLING SEQUENCE:
: 997 1982 1 FLUSH_FIDCACHE ()
: 998 1983 1
: 999 1984 1 INPUT PARAMETERS:
1000 1985 1 NONE
1001 1986 1
1002 1987 1 IMPLICIT INPUTS:
1003 1988 1 CURRENT_VCB: VCB of volume
1004 1989 1
1005 1990 1 OUTPUT PARAMETERS:
1006 1991 1 NONE
1007 1992 1
1008 1993 1 IMPLICIT OUTPUTS:
1009 1994 1 NONE
1010 1995 1
1011 1996 1 ROUTINE VALUE:
1012 1997 1 1
1013 1998 1
1014 1999 1 SIDE EFFECTS:
1015 2000 1 file ID cache cleared
1016 2001 1
1017 2002 1 --
1018 2003 1
1019 2004 2 BEGIN
1020 2005 2
1021 2006 2 BIND_COMMON;
1022 2007 2
1023 2008 2 LOCAL
1024 2009 2 FID_CACHE : REF BBLOCK; ! file ID cache
1025 2010 2
1026 2011 2
1027 2012 2 FID_CACHE = .BBLOCK [.CURRENT_VCB[VCB$$_CACHE], VCB$$_FIDCACHE];
1028 2013 2 FID_CACHE[VCB$$_FIDCOUNT] = 0;
1029 2014 2
1030 2015 2 1
: 1031 2016 1 END; ! end of routine FLUSH_FIDCACHE

```

					0000 00000 FLUSH_FIDCACHE:	
				.WORD	Save nothing	: 1971
50	98	AA	D0	00002	MOVL -104(BASE), R0	: 2012
50	58	B0	D0	00006	MOVL @88(R0), FID_CACHE	:
	02	A0	B4	0000A	CLRW 2(FID_CACHE)	: 2013
50		01	D0	0000D	MOVL #1, R0	: 2016

CLENUP
V04-000

I 13
16-Sep-1984 00:02:25
14-Sep-1984 12:30:12

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[F11X.SRC]CLENUP.B32;1 Page 28
(6)

04 00010

RET

;

; Routine Size: 17 bytes, Routine Base: \$CODE\$ + 0432


```

1033 2017 1 ROUTINE MAKE_DEACCESS : L_NORM =
1034 2018 1
1035 2019 1 ++
1036 2020 1
1037 2021 1 FUNCTIONAL DESCRIPTION:
1038 2022 1
1039 2023 1 This routine performs the machinery for deaccessing a file.
1040 2024 1
1041 2025 1 CALLING SEQUENCE:
1042 2026 1 MAKE_DEACCESS ()
1043 2027 1
1044 2028 1 INPUT PARAMETERS:
1045 2029 1 NONE
1046 2030 1
1047 2031 1 IMPLICIT INPUTS:
1048 2032 1 PRIMARY_FCB: FCB of file
1049 2033 1 CURRENT_WINDOW: window of file
1050 2034 1 CURRENT_VCB: VCB of volume in process
1051 2035 1
1052 2036 1 OUTPUT PARAMETERS:
1053 2037 1 NONE
1054 2038 1
1055 2039 1 IMPLICIT OUTPUTS:
1056 2040 1 NONE
1057 2041 1
1058 2042 1 ROUTINE VALUE:
1059 2043 1 NONE
1060 2044 1
1061 2045 1 SIDE EFFECTS:
1062 2046 1 file deaccessed
1063 2047 1
1064 2048 1 --
1065 2049 1
1066 2050 2 BEGIN
1067 2051 2
1068 2052 2 BIND_COMMON;
1069 2053 2
1070 2054 2 LOCAL
1071 2055 2 FCB : REF BBLOCK, ! local for primary fcb.
1072 2056 2 LCKMODE, ! lock mode for access lock.
1073 2057 2 WINDOW_SEGMENT : REF BBLOCK, ! address of the next window segment
1074 2058 2 DUMMY; ! dummy local to receive REMQUE
1075 2059 2
1076 2060 2 EXTERNAL
1077 2061 2 PMSSGL_OPEN : ADDRESSING_MODE (ABSOLUTE);
1078 2062 2 ! system count of currently open files
1079 2063 2
1080 2064 2 EXTERNAL ROUTINE
1081 2065 2 DEQ_LOCK : L_NORM, ! dequeue a lock
1082 2066 2 CONV_ACCLOCK : L_NORM, ! Convert file access lock.
1083 2067 2 LOCK_MODE : L_JSB_1ARG; ! Calculate access lock mode.
1084 2068 2
1085 2069 2 FCB = .PRIMARY_FCB;
1086 2070 2
1087 2071 2 ! Unlink the window from the FCB. Clear the applicable access conditions
1088 2072 2 ! in the FCB.
1089 2073 2

```

```

1090 2074 2
1091 2075 2 WINDOW_SEGMENT = .CURRENT_WINDOW;
1092 2076 2 DO
1093 2077 2 BEGIN
1094 2078 2 IF .WINDOW_SEGMENT[WCBSL_WFL] NEQ 0 THEN REMQUE (.WINDOW_SEGMENT, DUMMY);
1095 2079 2 WINDOW_SEGMENT = .WINDOW_SEGMENT[WCBSL_LINK];
1096 2080 2 END
1097 2081 2 UNTIL .WINDOW_SEGMENT EQL 0;
1098 2082 2
1099 2083 2 IF NOT .CURRENT_WINDOW [WCBSV_NOACCLOCK]
1100 2084 2 THEN
1101 2085 2 BEGIN
1102 2086 2 IF .CURRENT_WINDOW[WCBSV_NOREAD]
1103 2087 2 THEN FCB[FCBSV_EXCL] = 0;
1104 2088 2
1105 2089 2 IF .CURRENT_WINDOW[WCBSV_NOTRUNC]
1106 2090 2 THEN FCB[FCBSW_TCNT] = .FCB[FCBSW_TCNT] - 1;
1107 2091 2
1108 2092 2 IF .CURRENT_WINDOW[WCBSV_NOWRITE]
1109 2093 2 THEN FCB[FCBSW_LCNT] = .FCB[FCBSW_LCNT] - 1;
1110 2094 2
1111 2095 2 FCB [FCBSW_ACNT] = .FCB [FCBSW_ACNT] - 1;
1112 2096 2
1113 2097 2 END;
1114 2098 2 ! of normal (not NOLOCK) deaccess.
1115 2099 2 FCB[FCBSW_REFCNT] = .FCB[FCBSW_REFCNT] - 1;
1116 2100 2
1117 2101 2 ! For a write access, bump down the writer count. If this is the
1118 2102 2 ! last write, and the file is the index file or the storage map, clear
1119 2103 2 ! the appropriate flag in the VCB. If there's a cache lock being held
1120 2104 2 ! for this file, release it.
1121 2105 2 !
1122 2106 2
1123 2107 2 IF .CURRENT_WINDOW[WCBSV_WRITE]
1124 2108 2 THEN
1125 2109 2 BEGIN
1126 2110 2
1127 2111 2 IF NOT .CURRENT_WINDOW [WCBSV_NOACCLOCK]
1128 2112 2 THEN
1129 2113 2 FCB[FCBSW_WCNT] = .FCB[FCBSW_WCNT] - 1;
1130 2114 2
1131 2115 2 IF .FCB[FCBSW_WCNT] EQL 0
1132 2116 2 OR (.FCB [FCBSW_REFCNT] EQL 0 AND .CURRENT_WINDOW [WCBSV_WRITE])
1133 2117 2 THEN
1134 2118 2 BEGIN
1135 2119 2 IF .FCB[FCBSB_FID_NMX] EQL 0
1136 2120 2 THEN
1137 2121 2 BEGIN
1138 2122 2 IF .FCB[FCBSW_FID_NUM] EQL 1
1139 2123 2 THEN CURRENT_VCB[VCBSV_WRITE_IF] = 0;
1140 2124 2 IF .FCB[FCBSW_FID_NUM] EQL 2
1141 2125 2 THEN CURRENT_VCB[VCBSV_WRITE_SM] = 0;
1142 2126 2 END;
1143 2127 2 IF .FCB[FCBSL_CACHELKID] NEQ 0
1144 2128 2 THEN
1145 2129 2 BEGIN
1146 2130 2 DEQ_LOCK (.FCB[FCBSL_CACHELKID]);

```



```

1147 2131 5          FCB[FCB$$_CACHELKID] = 0;
1148 2132 4          END;
1149 2133 3          END;
1150 2134 2          END;
1151 2135 2          ! Recalculate the lock mode of the access lock for this fcb.
1152 2136 2          !
1153 2137 2          !
1154 2138 2          IF .FCB [FCB$_ACNT] EQL 0
1155 2139 2          THEN
1156 2140 2              LCKMODE = LCK$_NLMODE
1157 2141 2          ELSE
1158 2142 2              BEGIN
1159 2143 2                  LOCAL
1160 2144 2                      ACCTL;
1161 2145 2                  ACCTL = 0;
1162 2146 2                  IF .FCB [FCB$_WCNT] NEQ 0
1163 2147 2                      THEN ACCTL = .ACCTL + FIB$_WRITE;
1164 2148 2                  IF .FCB [FCB$_LCNT] NEQ 0
1165 2149 2                      THEN ACCTL = .ACCTL + FIB$_NOWRITE;
1166 2150 2                  LCKMODE = LOCK_MODE (.ACCTL);
1167 2151 2              END;
1168 2152 2          ! If the new access lock mode lock for this fcb is different (lower)
1169 2153 2          ! than the current lock, convert it. The conversion routine will also
1170 2154 2          ! dequeue the lock if this is the last reference.
1171 2155 2          !
1172 2156 2          IF .LCKMODE<0,8> NEQ .FCB [FCB$_ACCLCKMODE]
1173 2157 2          OR .FCB [FCB$_REFCNT] EQL 0
1174 2158 2          THEN
1175 2159 2              IF NOT CONV_ACCLOCK (.LCKMODE, .FCB)
1176 2160 2              THEN
1177 2161 2                  BUG_CHECK (XQPERR, 'deaccess conversion failed');
1178 2162 2          ! Note: We now have a file control block with a possible zero access count
1179 2163 2          ! in the FCB list. This gets dealt with by the general cleanup.
1180 2164 2          !
1181 2165 2          PMSSGL_OPEN = .PMSSGL_OPEN - 1;          ! bump down count of open files
1182 2166 2          CURRENT_VCB[VCB$_TRANS] = .CURRENT_VCB[VCB$_TRANS] - 1;
1183 2167 2          !
1184 2168 2          !
1185 2169 2          !
1186 2170 2          !
1187 2171 2          !
1188 2172 2          !
1189 2173 2          !
1190 2174 2          !
1191 2175 2          !
1192 2176 2          !
1193 2177 2          !
1194 2178 1          END;

```

! end of routine MAKE_DEACCESS

```

.EXTRN PMSSGL_OPEN, DEQ_LOCK
.EXTRN CONV_ACCLOCK, LOCK_MODE
.EXTRN BUG$_XQPERR

```

000C 00000 MAKE_DEACCESS:

.WORD Save R2,R3

; 2017

		51	0C	AA	9E	00002	MOVAB	12(BASE), R1	2050	
		52	08	AA	D0	00006	MOVL	8(BASE), FCB	2069	
		50		61	D0	0000A	MOVL	(R1), WINDOW_SEGMENT	2075	
				60	D5	0000D	1\$: TSTL	(WINDOW_SEGMENT)	2078	
				03	13	0000F	BEQL	2\$		
		53		60	0F	00011	REMQUE	(WINDOW_SEGMENT), DUMMY		
		50	20	A0	D0	00014	2\$: MOVL	32(WINDOW_SEGMENT), WINDOW_SEGMENT	2079	
				F3	12	00018	BNEQ	1\$	2081	
		50		61	D0	0001A	MOVL	(R1), R0	2083	
21	14	A0		02	E0	0001D	BBS	#2, 20(R0), 6\$		
04	15	A0		02	E1	00022	BBC	#2, 21(R0), 3\$	2086	
	22	A2		08	8A	00027	BICB2	#8, 34(FCB)	2087	
		50		61	D0	0002B	3\$: MOVL	(R1), R0	2089	
03	15	A0		03	E1	0002E	BBC	#3, 21(R0), 4\$		
			20	A2	B7	00033	DECW	32(FCB)	2090	
		50		61	D0	00036	4\$: MOVL	(R1), R0	2092	
		03		14	A0	E9	00039	BLBC	20(R0), 5\$	
			1E	A2	B7	0003D	DECW	30(FCB)	2093	
			1A	A2	B7	00040	5\$: DECW	26(FCB)	2095	
			18	A2	B7	00043	6\$: DECW	24(FCB)	2099	
		50		61	D0	00046	MOVL	(R1), R0	2107	
4B	0B	A0		01	E1	00049	BBC	#1, 11(R0), 11\$		
03	14	A0		02	E0	0004E	BBS	#2, 20(R0), 7\$	2111	
			1C	A2	B7	00053	DECW	28(FCB)	2113	
			1C	A2	B5	00056	7\$: TSTW	28(FCB)	2115	
				0D	13	00059	BEQL	8\$		
			18	A2	B5	0005B	TSTW	24(FCB)	2116	
				39	12	0005E	BNEQ	11\$		
		50		61	D0	00060	MOVL	(R1), R0		
31	0B	A0		01	E1	00063	BBC	#1, 11(R0), 11\$		
			29	A2	95	00068	8\$: TSTB	41(FCB)	2119	
				1C	12	0006B	BNEQ	10\$		
		01	24	A2	B1	0006D	CMPW	36(FCB), #1	2122	
				08	12	00071	BNEQ	9\$		
		50		98	AA	D0	00073	MOVL	-104(BASE), R0	2123
	0B	A0		01	8A	00077	BICB2	#1, 11(R0)		
		02	24	A2	B1	0007B	9\$: CMPW	36(FCB), #2	2124	
				08	12	0007F	BNEQ	10\$		
		50		98	AA	D0	00081	MOVL	-104(BASE), R0	2125
	0B	A0		02	8A	00085	BICB2	#2, 11(R0)		
			54	A2	D5	00089	10\$: TSTL	84(FCB)	2127	
				0B	13	0008C	BEQL	11\$		
			54	A2	DD	0008E	PUSHL	84(FCB)	2130	
	0000G	CF		01	FB	00091	CALLS	#1, DEQ_LOCK		
			54	A2	D4	00096	CLRL	84(FCB)	2131	
			1A	A2	B5	00099	11\$: TSTW	26(FCB)	2139	
				04	12	0009C	BNEQ	12\$		
				51	D4	0009E	CLRL	LCKMODE	2141	
				19	11	000A0	BRB	15\$		
				50	D4	000A2	12\$: CLRL	ACCTL	2147	
			1C	A2	B5	000A4	TSTW	28(FCB)	2148	
				05	13	000A7	BEQL	13\$		
		50	0100	C0	9E	000A9	MOVAB	256(R0), ACCTL	2149	
			1E	A2	B5	000AE	13\$: TSTW	30(FCB)	2150	
				02	13	000B1	BEQL	14\$		
				50	D6	000B3	INCL	ACCTL	2151	
				0000G	30	000B5	14\$: BSBW	LOCK_MODE	2153	

CLENUP
V04-000

N 13
16-Sep-1984 00:02:25
14-Sep-1984 12:30:12

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[F11X.SRC]CLENUP.B32;1

Page 33
(7)

0B	51	50	D0	000B8	15\$:	MOVL	R0, LCKMODE	:	2162
	A2	51	91	000BB		CMPB	LCKMODE, 11(FCB)	:	
		05	12	000BF		BNEQ	16\$:	
		18	A2	B5	000C1	TSTW	24(FCB)	:	2163
			0E	12	000C4	BNEQ	17\$:	
0000G	CF	06	BB	000C6	16\$:	PUSHR	#^M<R1,R2>	:	2165
	04	02	FB	000C8		CALLS	#2, CONV_ACCLOCK	:	
		50	E8	000CD		BLBS	R0, 17\$:	
			FEFF	000D0		BUGW		:	2167
			0000*	000D2		.WORD	<BUG\$ XQPERR!4>	:	
	00000000G	9F	D7	000D4	17\$:	DECL	@#PMS\$GL_OPEN	:	2173
	50	98	AA	D0	000DA	MOVL	-104(BASE), R0	:	2174
		0C	A0	B7	000DE	DECW	12(R0)	:	
	50		01	D0	000E1	MOVL	#1, R0	:	2176
			04	000E4		RET		:	2178

; Routine Size: 229 bytes, Routine Base: \$CODE\$ + 0443

```
: 1196 2179 1 GLOBAL ROUTINE DEL_EXTFCB (START_FCB) : L_NORM =
: 1197 2180 1
: 1198 2181 1 ++
: 1199 2182 1
: 1200 2183 1 FUNCTIONAL DESCRIPTION:
: 1201 2184 1
: 1202 2185 1 This routine removes and deallocates all extension FCB's, if any,
: 1203 2186 1 linked to the indicated FCB.
: 1204 2187 1
: 1205 2188 1 CALLING SEQUENCE:
: 1206 2189 1 DEL_EXTFCB (ARG1)
: 1207 2190 1
: 1208 2191 1 INPUT PARAMETERS:
: 1209 2192 1 ARG1: address of primary FCB or 0
: 1210 2193 1
: 1211 2194 1 IMPLICIT INPUTS:
: 1212 2195 1 NONE
: 1213 2196 1
: 1214 2197 1 OUTPUT PARAMETERS:
: 1215 2198 1 NONE
: 1216 2199 1
: 1217 2200 1 IMPLICIT OUTPUTS:
: 1218 2201 1 NONE
: 1219 2202 1
: 1220 2203 1 ROUTINE VALUE:
: 1221 2204 1 NONE
: 1222 2205 1
: 1223 2206 1 SIDE EFFECTS:
: 1224 2207 1 FCB's deallocated
: 1225 2208 1
: 1226 2209 1 --
: 1227 2210 1
: 1228 2211 2 BEGIN
: 1229 2212 2
: 1230 2213 2 MAP
: 1231 2214 2 START_FCB : REF BBLOCK; ! FCB argument
: 1232 2215 2
: 1233 2216 2 LOCAL
: 1234 2217 2 FCB : REF BBLOCK, ! running FCB pointer
: 1235 2218 2 NEXT_FCB : REF BBLOCK, ! next extension FCB
: 1236 2219 2 P : REF BBLOCK, ! pointer to chase for VCB
: 1237 2220 2 DUMMY; ! dummy local to receive REMQUE
: 1238 2221 2
: 1239 2222 2 BASE_REGISTER;
: 1240 2223 2
: 1241 2224 2 EXTERNAL ROUTINE
: 1242 2225 2 DEALLOCATE : L_NORM; ! deallocate dynamic memory
: 1243 2226 2
: 1244 2227 2 ! Checking for null pointers, find the first extension FCB. Follow the extension
: 1245 2228 2 ! list and remove and deallocate the extension FCB's, cleaning out the pointers
: 1246 2229 2 ! on the way. For each FCB removed, we must find the VCB (by chasing around the
: 1247 2230 2 ! FCB list) and decrement the transaction count.
: 1248 2231 2
: 1249 2232 2
: 1250 2233 2 IF .START_FCB EQL 0 THEN RETURN 1;
: 1251 2234 2 FCB = .START_FCB[FCB$L_EXFCB];
: 1252 2235 2 START_FCB[FCB$L_EXFCB] = 0;
```



```
: 1253      2236 2 UNTIL .FCB EQL 0 DO
: 1254      2237 BEGIN
: 1255      2238 NEXT_FCB = .FCB[FCB$L_EXFCB];
: 1256      2239
: 1257      2240 P = .FCB[FCB$L_FCBFL];
: 1258      2241 UNTIL .P[VCB$B_TYPE] EQL DYN$C_VCB
: 1259      2242 DO P = .P[FCB$C_FCBFL];
: 1260      2243 P[VCB$W_TRANS] = .P[VCB$W_TRANS] - 1;
: 1261      2244
: 1262      2245 FCB[FCB$L_EXFCB] = 0;
: 1263      2246 IF .FCB [FCB$B_TYPE] NEQ DYN$C_FCB
: 1264      2247 THEN
: 1265      2248     BUG CHECK (NOTFCBFCB, 'not fcb');
: 1266      2249     REMQUE T.FCB, DUMMY);
: 1267      2250     DEALLOCATE (.FCB);
: 1268      2251     FCB = .NEXT_FCB;
: 1269      2252     END;
: 1270      2253
: 1271      2254 RETURN 1;
: 1272      2255
: 1273      2256 1 END;
```

! end of routine DEL_EXTFCB

				.EXTRN	BUG\$_NOTFCBFCB	
				.ENTRY	DEL_EXTFCB, Save R2,R3,R4,R5	: 2179
50	04	AC	D0 00002	MOVL	START_FCB, R0	: 2233
		3C	13 00006	BEQL	5\$	
53	0C	A0	D0 00008	MOVL	12(R0), FCB	: 2234
	0C	A0	D4 0000C	CLRL	12(R0)	: 2235
		53	D5 0000F 1\$:	TSTL	FCB	: 2236
		31	13 00011	BEQL	5\$	
54	0C	A3	D0 00013	MOVL	12(FCB), NEXT_FCB	: 2238
52		63	D0 00017	MOVL	(FCB), P	: 2240
11	0A	A2	91 0001A 2\$:	CMPB	10(P), #17	: 2241
		05	13 0001E	BEQL	3\$	
52		62	D0 00020	MOVL	(P), P	: 2242
		F5	11 00023	BRB	2\$	
	0C	A2	B7 00025 3\$:	DECW	12(P)	: 2243
	0C	A3	D4 00028	CLRL	12(FCB)	: 2245
07	0A	A3	91 0002B	CMPB	10(FCB), #7	: 2246
		04	13 0002F	BEQL	4\$	
			FEFF 00031	BUGW		: 2248
			0000* 00033	.WORD	<BUG\$_NOTFCBFCB!4>	
55		63	0F 00035 4\$:	REMQUE	(FCB), DUMMY	: 2249
		53	DD 00038	PUSHL	FCB	: 2250
0000G	CF	01	FB 0003A	CALLS	#1, DEALLOCATE	
53		54	D0 0003F	MOVL	NEXT_FCB, FCB	: 2251
		CB	11 00042	BRB	1\$: 2236
50		01	D0 00044 5\$:	MOVL	#1, R0	: 2254
		04	00047	RET		: 2256

; Routine Size: 72 bytes, Routine Base: \$CODE\$ + 0528


```

: 1275 2257 1 ROUTINE ZERO_CHANNEL : L_NORM =
: 1276 2258 1
: 1277 2259 1 ++
: 1278 2260 1
: 1279 2261 1 FUNCTIONAL DESCRIPTION:
: 1280 2262 1
: 1281 2263 1 This routine zeroes out the window pointer being returned to
: 1282 2264 1 the user for his channel control block. It also credits one to the
: 1283 2265 1 user's open file quota, except for the case of a shared window.
: 1284 2266 1 This routine must be executed in kernel mode.
: 1285 2267 1
: 1286 2268 1 CALLING SEQUENCE:
: 1287 2269 1 ZERO_CHANNEL ()
: 1288 2270 1
: 1289 2271 1 INPUT PARAMETERS:
: 1290 2272 1 NONE
: 1291 2273 1
: 1292 2274 1 IMPLICIT INPUTS:
: 1293 2275 1 IO_PACKET: I/O packet of request
: 1294 2276 1
: 1295 2277 1 OUTPUT PARAMETERS:
: 1296 2278 1 NONE
: 1297 2279 1
: 1298 2280 1 IMPLICIT OUTPUTS:
: 1299 2281 1 NONE
: 1300 2282 1
: 1301 2283 1 ROUTINE VALUE:
: 1302 2284 1 NONE
: 1303 2285 1
: 1304 2286 1 SIDE EFFECTS:
: 1305 2287 1 channel window pointer cleared, file quota bumped unless shared window
: 1306 2288 1
: 1307 2289 1 --
: 1308 2290 1
: 1309 2291 2 BEGIN
: 1310 2292 2
: 1311 2293 2 LOCAL
: 1312 2294 2 ABD : REF BBLOCKVECTOR [,ABD$C_LENGTH],
: 1313 2295 2 : ! buffer descriptors
: 1314 2296 2 JIB : REF BBLOCK, : Job information block address
: 1315 2297 2 PCB : REF BBLOCK; : address of user process control block
: 1316 2298 2
: 1317 2299 2 EXTERNAL
: 1318 2300 2 SCH$GL_PCBVEC : REF VECTOR ADDRESSING_MODE (ABSOLUTE);
: 1319 2301 2 : ! system PCB vector
: 1320 2302 2
: 1321 2303 2 BIND_COMMON;
: 1322 2304 2
: 1323 2305 2 ! pointer to buffer descriptors
: 1324 2306 2 ABD = .BBLOCK [,IO_PACKET[IRP$L_SVAPTE], AIB$L_DESCRIPTOR];
: 1325 2307 2 ABD[ABD$C_WINDOW, ABD$W_COUNT] = 4;
: 1326 2308 2 .ABD[ABD$C_WINDOW, ABD$W_TEXT] + ABD[ABD$C_WINDOW, ABD$W_TEXT] + 1 = 0;
: 1327 2309 2
: 1328 2310 2 IF
: 1329 2311 2 BEGIN
: 1330 2312 2
: 1331 2313 2 ! The FILCNT quota is credited if a WCB has not yet been allocated or
```



```
: 1332      2314 3      ! if the SHRWCB bit is not set in the WCB.
: 1333      2315 3
: 1334      2316 3      IF .CURRENT_WINDOW EQL 0
: 1335      2317 3      THEN 1
: 1336      2318 3      ELSE NOT .CURRENT_WINDOW[WCBSV_SHRWCB]
: 1337      2319 3      END
: 1338      2320 3      THEN
: 1339      2321 3      BEGIN
: 1340      2322 3      PCB = .SCH$GL_PCBVEC[(IO_PACKET[IRPSL_PID])<0,16>];
: 1341      2323 3      JIB = .PCB[PCBSL_JIB];
: 1342      2324 3      JIB[JIB$W_FILCNT] = .JIB[JIB$W_FILCNT] + 1;
: 1343      2325 3      END;
: 1344      2326 3
: 1345      2327 3      RETURN 1;
: 1346      2328 3
: 1347      2329 1      END;
```

! end of routine ZERO_CHANNEL

```
.EXTRN SCH$GL_PCBVEC

0000 00000 ZERO_CHANNEL:
      50      90  AA  D0 00002      .WORD      Save nothing      : 2257
      51      2C  B0  D0 00006      MOVL      -112(BASE), R0      : 2306
02    A1      04  B0 0000A      MOVW      #4, 2(ABD)      : 2307
      50      61  3C 0000E      MOVZWL     (ABD), R0      : 2308
      01 A140  9F 00011      PUSHAB     1(ABD)[R0]
      9E  D4 00015      CLRL      @(SP)+
      50      0C  AA  D0 00017      MOVL      12(BASE), R0      : 2316
      05  13 0001B      BEQL      1$
1D    0B    A0      03  E0 0001D      BBS      #3, 11(R0), 2$      : 2318
      51 00000000G  9F  D0 00022 1$:      MOVL      @#SCH$GL_PCBVEC, R1      : 2322
      50      90  AA  D0 00029      MOVL      -112(BASE), R0
      50      0C  C0 0002D      ADDL2     #12, R0
      50      60  3C 00030      MOVZWL     (R0), R0
      50      6140 D0 00033      MOVL      (R1)[R0], PCB
      50      0080 C0  D0 00037      MOVL      128(PCB), JIB
      50      30  A0  B6 0003C      INCW      48(JIB)
      50      01  D0 0003F 2$:      MOVL      #1, R0
      04 00042      RET
```

; Routine Size: 67 bytes, Routine Base: \$CODE\$ + 0570

```
: 1349 2330 1 GLOBAL ROUTINE NUKE_HEAD_FCB (FCB) : L_NORM NOVALUE =
: 1350 2331 1
: 1351 2332 1 !++
: 1352 2333 1
: 1353 2334 1 Functional Description:
: 1354 2335 1
: 1355 2336 1 Given an fcb already stripped of possible extension fcbs,
: 1356 2337 1 and which has a refcnt of 0 (assumed), clean up the things
: 1357 2338 1 that need cleaning up, remove it from the fcb list (we assume
: 1358 2339 1 that is where it is), and deallocate it.
: 1359 2340 1
: 1360 2341 1 --
: 1361 2342 1
: 1362 2343 2 BEGIN
: 1363 2344 2
: 1364 2345 2 MAP
: 1365 2346 2 FCB : REF BBLOCK;
: 1366 2347 2
: 1367 2348 2 BASE_REGISTER;
: 1368 2349 2
: 1369 2350 2 EXTERNAL ROUTINE
: 1370 2351 2 ACL_DELETEACL,
: 1371 2352 2 CONV_ACCLOCK : L_NORM,
: 1372 2353 2 DEALLOCATE : L_NORM;
: 1373 2354 2
: 1374 2355 2 LOCAL
: 1375 2356 2 DUMMY;
: 1376 2357 2
: 1377 2358 2 IF .FCB [FCB$B_TYPE] NEQ DYN$C_FCB
: 1378 2359 2 THEN
: 1379 2360 2 BUG_CHECK (NOTFCBFCB, 'not fcb');
: 1380 2361 2
: 1381 2362 2 REMQUE (.FCB, DUMMY);
: 1382 2363 2
: 1383 2364 2 IF .BBLOCK [FCB [FCB$R_ORB], ORB$V_ACL_QUEUE]
: 1384 2365 2 THEN
: 1385 2366 2 ACL_DELETEACL (FCB [FCB$L_ACLFL], 0);
: 1386 2367 2
: 1387 2368 2 IF NOT CONV_ACCLOCK (0, .FCB)
: 1388 2369 2 THEN
: 1389 2370 2 BUG_CHECK (XQPERR, 'Unexpected lock manager status');
: 1390 2371 2
: 1391 2372 2 DEALLOCATE (.FCB);
: 1392 2373 2
: 1393 2374 1 END; ! of routine NUKE_HEAD_FCB
```

				.EXTRN	ACL_DELETEACL	
				.ENTRY	NUKE_HEAD_FCB, Save nothing	: 2330
50	04	AC	D0	MOVL	FCB, R0	: 2358
07	0A	A0	91	CMPB	10(R0), #7	
		04	13	BEQL	1\$	
				BUGW		: 2360
				.WORD	<BUG\$_NOTFCBFCB!4>	
50	04	BC	0F	REMQUE	@FCB, DUMMY	: 2362

CLENUP
V04-000

G 14
16-Sep-1984 00:02:25
14-Sep-1984 12:30:12

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[F11X.SRC]CLENUP.B32;1

Page 39
(10)

10	63	50	04	AC	D0	00014	MOVL	FCB, R0	:	2364
		A0		01	E1	00018	BBC	#1, 99(R0), 2\$:	
7E	04	AC	00000080	7E	D4	0001D	CLRL	-(SP)	:	2366
	0000G	CF		8F	C1	0001F	ADDL3	#128, FCB, -(SP)	:	
			04	02	FB	00028	CALLS	#2, ACL_DELETEACL	:	
				AC	DD	0002D	PUSHL	FCB	:	2368
	0000G	CF		7E	D4	00030	CLRL	-(SP)	:	
		04		02	FB	00032	CALLS	#2, CONV_ACCLOCK	:	
				50	E8	00037	BLBS	R0, 3\$:	
					FEFF	0003A	BUGW		:	2370
					0000*	0003C	.WORD	<BUG\$_XQPERR!4>	:	
	0000G	CF	04	AC	DD	0003E	PUSHL	FCB	:	2372
				01	FB	00041	CALLS	#1, DEALLOCATE	:	
					04	00046	RET		:	2374

; Routine Size: 71 bytes, Routine Base: \$CODE\$ + 05B3

CP
VO
:
:

```
.PSECT $LOCKEDC1$,NOWRT,2
```

PC	Op	OpC	OpD	OpI	OpR	OpS	OpT	OpV	OpW	OpX	OpY	OpZ	OpAA	OpAB	OpAC	OpAD	OpAE	OpAF	OpAG	OpAH	OpAI	OpAJ	OpAK	OpAL	OpAM	OpAN	OpAO	OpAP	OpAQ	OpAR	OpAS	OpAT	OpAU	OpAV	OpAW	OpAX	OpAY	OpAZ	OpBA	OpBB	OpBC	OpBD	OpBE	OpBF	OpBG	OpBH	OpBI	OpBJ	OpBK	OpBL	OpBM	OpBN	OpBO	OpBP	OpBQ	OpBR	OpBS	OpBT	OpBU	OpBV	OpBW	OpBX	OpBY	OpBZ	OpCA	OpCB	OpCC	OpCD	OpCE	OpCF	OpCG	OpCH	OpCI	OpCJ	OpCK	OpCL	OpCM	OpCN	OpCO	OpCP	OpCQ	OpCR	OpCS	OpCT	OpCU	OpCV	OpCW	OpCX	OpCY	OpCZ	OpDA	OpDB	OpDC	OpDD	OpDE	OpDF	OpDG	OpDH	OpDI	OpDJ	OpDK	OpDL	OpDM	OpDN	OpDO	OpDP	OpDQ	OpDR	OpDS	OpDT	OpDU	OpDV	OpDW	OpDX	OpDY	OpDZ	OpEA	OpEB	OpEC	OpED	OpEE	OpEF	OpEG	OpEH	OpEI	OpEJ	OpEK	OpEL	OpEM	OpEN	OpEO	OpEP	OpEQ	OpER	OpES	OpET	OpEU	OpEV	OpEW	OpEX	OpEY	OpEZ	OpFA	OpFB	OpFC	OpFD	OpFE	OpFF	OpFG	OpFH	OpFI	OpFJ	OpFK	OpFL	OpFM	OpFN	OpFO	OpFP	OpFQ	OpFR	OpFS	OpFT	OpFU	OpFV	OpFW	OpFX	OpFY	OpFZ	OpGA	OpGB	OpGC	OpGD	OpGE	OpGF	OpGG	OpGH	OpGI	OpGJ	OpGK	OpGL	OpGM	OpGN	OpGO	OpGP	OpGQ	OpGR	OpGS	OpGT	OpGU	OpGV	OpGW	OpGX	OpGY	OpGZ	OpHA	OpHB	OpHC	OpHD	OpHE	OpHF	OpHG	OpHH	OpHI	OpHJ	OpHK	OpHL	OpHM	OpHN	OpHO	OpHP	OpHQ	OpHR	OpHS	OpHT	OpHU	OpHV	OpHW	OpHX	OpHY	OpHZ	OpIA	OpIB	OpIC	OpID	OpIE	OpIF	OpIG	OpIH	OpII	OpIJ	OpIK	OpIL	OpIM	OpIN	OpIO	OpIP	OpIQ	OpIR	OpIS	OpIT	OpIU	OpIV	OpIW	OpIX	OpIY	OpIZ	OpJA	OpJB	OpJC	OpJD	OpJE	OpJF	OpJG	OpJH	OpJI	OpJJ	OpJK	OpJL	OpJM	OpJN	OpJO	OpJP	OpJQ	OpJR	OpJS	OpJT	OpJU	OpJV	OpJW	OpJX	OpJY	OpJZ	OpKA	OpKB	OpKC	OpKD	OpKE	OpKF	OpKG	OpKH	OpKI	OpKJ	OpKK	OpKL	OpKM	OpKN	OpKO	OpKP	OpKQ	OpKR	OpKS	OpKT	OpKU	OpKV	OpKW	OpKX	OpKY	OpKZ	OpLA	OpLB	OpLC	OpLD	OpLE	OpLF	OpLG	OpLH	OpLI	OpLJ	OpLK	OpLL	OpLM	OpLN	OpLO	OpLP	OpLQ	OpLR	OpLS	OpLT	OpLU	OpLV	OpLW	OpLX	OpLY	OpLZ	OpMA	OpMB	OpMC	OpMD	OpME	OpMF	OpMG	OpMH	OpMI	OpMJ	OpMK	OpML	OpMM	OpMN	OpMO	OpMP	OpMQ	OpMR	OpMS	OpMT	OpMU	OpMV	OpMW	OpMX	OpMY	OpMZ	OpNA	OpNB	OpNC	OpND	OpNE	OpNF	OpNG	OpNH	OpNI	OpNJ	OpNK	OpNL	OpNM	OpNN	OpNO	OpNP	OpNQ	OpNR	OpNS	OpNT	OpNU	OpNV	OpNW	OpNX	OpNY	OpNZ	OpOA	OpOB	OpOC	OpOD	OpOE	OpOF	OpOG	OpOH	OpOI	OpOJ	OpOK	OpOL	OpOM	OpON	OpOO	OpOP	OpOQ	OpOR	OpOS	OpOT	OpOU	OpOV	OpOW	OpOX	OpOY	OpOZ	OpPA	OpPB	OpPC	OpPD	OpPE	OpPF	OpPG	OpPH	OpPI	OpPJ	OpPK	OpPL	OpPM	OpPN	OpPO	OpPP	OpPQ	OpPR	OpPS	OpPT	OpPU	OpPV	OpPW	OpPX	OpPY	OpPZ	OpQA	OpQB	OpQC	OpQD	OpQE	OpQF	OpQG	OpQH	OpQI	OpQJ	OpQK	OpQL	OpQM	OpQN	OpQO	OpQP	OpQQ	OpQR	OpQS	OpQT	OpQU	OpQV	OpQW	OpQX	OpQY	OpQZ	OpRA	OpRB	OpRC	OpRD	OpRE	OpRF	OpRG	OpRH	OpRI	OpRJ	OpRK	OpRL	OpRM	OpRN	OpRO	OpRP	OpRQ	OpRR	OpRS	OpRT	OpRU	OpRV	OpRW	OpRX	OpRY	OpRZ	OpSA	OpSB	OpSC	OpSD	OpSE	OpSF	OpSG	OpSH	OpSI	OpSJ	OpSK	OpSL	OpSM	OpSN	OpSO	OpSP	OpSQ	OpSR	OpSS	OpST	OpSU	OpSV	OpSW
----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

CLENUP
V04-000

I 14
16-Sep-1984 00:02:25
14-Sep-1984 12:30:12

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[F11X.SRC]CLENUP.B32;1 (11)

Page 41

05 00017

RSB

; Routine Size: 24 bytes, Routine Base: \$LOCKEDC1\$ + 0000

```
: 1436      2416 1
: 1437      2417 1
: 1438      2418 1
: 1439      2419 1
: 1440      2420 1
: 1441      2421 1
: 1442      2422 1
: 1443      2423 1
: 1444      2424 0
:           0 ELUDOM
```

Note that just prior to the SET_DIRINDX routine the psects were changed to the locked psect because the SET_DIRINDX routine must be locked. Any routines added at this point will be locked also, so unless they need to be locked, put them prior to SET_DIRINDX.

PSECT SUMMARY

Name	Bytes	Attributes
\$CODE\$	1530 NOVEC,NOWRT, RD	EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
\$LOCKEDC1\$	24 NOVEC,NOWRT, RD	EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	95	0	1000	00:02.0

COMMAND QUALIFIERS

; BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:CLENUP/OBJ=OBJ\$:CLENUP MSRC\$:CLENUP/UPDATE=(ENH\$:CLENUP)

```
: Size:      1554 code + 0 data bytes
: Run Time:   01:19.3
: Elapsed Time: 02:31.2
: Lines/CPU Min: 1834
: Lexemes/CPU-Min: 54610
: Memory Used: 371 pages
: Compilation Complete
```


0168

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY